MSX

Praktijkprogramma's 1

Wessel Akkermans

Wessel Akkermans

MSX
Praktijkprogramma's

uw **MSX** *computer de baas*

Beste lezer,

Dit MSX boek is afkomstig uit de nalatenschap van Wammes Witkop - hoofdredacteur MSX Computer Magazine 1985 - 1992.

MSX praktijkprogramma's
deel 1

MSX

Praktijkprogramma's

deel 1

uw **MSX** *computer*
de baas

Wessel Akkermans

uitgeverij STARK - TEXEL
postbus 302 - 1794 ZG Oosterend tel. 02223 - 661

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Akkermans, Wessel

MSX praktijkprogramma's : uw MSX computer de baas /
Wessel Akkermans. – Oosterend : Stark-Texel
ISBN 90-6398-437-5
SISO 365.3 UDC 681.3.06
Trefw.: MSX (computer) / microcomputers ; programmeren.

. . . .

april 1985
ISBN 90 6398 437 5

Inleiding

Wie zelf zijn MSX-computer wil leren programmeren, zal dat met behulp van het bij de computer meegeleverde handboek moeten doen. Eventueel kan hij zich nog een Nederlandstalig handboek aanschaffen, of een basis-cursus BASIC volgen. Het nadeel van al deze boeken is echter dat ze niet veel verder gaan dan het doen van een opgave van de in de computer aanwezige BASIC-statements en functies, en het opsommen van wat deze statements en functies wel en niet kunnen. In een BASIC-cursus leert men, mits de cursus goed is, ook nog hoe men de statements moet toepassen. Een basiscursus BASIC behandelt echter slechts een beperkt aantal, meest gangbare, BASIC-statements. Dit alles heeft tot gevolg dat men een moeizame weg moet bewandelen om het BASIC van een bepaalde computer onder de knie te krijgen.

Wie voldoende basiskennis over het BASIC van zijn computer heeft opgedaan, die kan een of meer boeken met kant en klare toepassingsprogramma's kopen. Met behulp van die boeken kan hij dan proberen zijn theoretische kennis aan de praktijk te toetsen. In deze boeken staan vaak verbluffend slimme en handige programma's Daarentegen worden van die programma's meestal ook niet meer dan de listings gegeven. Het doorgronden van de programma's wordt geheel aan de lezer over gelaten. Het is mijn ervaring dat het vaak erg moeilijk is om, zonder uitleg van de programmeur, na te gaan wat een programma precies doet. Vaak zal een programmeur om geheugenruimte te sparen of om de snelheid te verhogen, bepaalde statements in een bepaalde volgorde gebruiken, met als gevolg dat het programma beter functioneert, doch dat het tegelijkertijd ondoorgrondelijker wordt.

Alle hiervoor genoemde feiten hebben mij er toe gebracht een aantal programma's en programmeertips op een andere manier naar voren te brengen. In dit boek zal ik proberen u duidelijk te maken hoe je bepaalde statements kunt toepassen. Dit zal ik doen aan de hand van een aantal probleemstellingen, een analyse van dat probleem en ten slotte de oplossing in de vorm van een programma. Mijn doel daarbij is te bereiken dat u in staat zult zijn de gegeven programma's naar uw eigen inzicht en behoefte aan te passen. Het zal u dan ook opvallen (zeker de meer ervaren programmeurs onder u) dat ik vaak niet de meest effectieve oplossing heb gekozen. Voor dit boek vind ik het belangrijker dat de oplossing worden begrepen, dan dat ze op de meest effectieve manier door de computer worden uitgevoerd.

Over de in dit boek gegeven listings en voorbeelden van de resultaten van het uitvoeren van de programma's moeten mij nog enkele opmerkingen van het hart. Alle listings en voorbeelden zijn gemaakt op de aan mijn MSX-computer aangesloten printer, een Brother CE50BT. Deze listings zijn rechtstreeks overgenomen in dit boek om eventuele zetfouten te voorkomen. In deze listings zijn, om de leesbaarheid te verhogen, REM-statements opgenomen. In MSX-BASIC zijn er twee manieren om aan te geven dat een regel een REM-regel is; d.m.v. het REM-statement of d.m.v. een apostrof ('). Beide manieren zijn in de listings toegepast. REM-regels mag u echter zonder meer weglaten. Dit scheelt u typewerk, en het scheelt geheugenruimte in de computer. Bovendien zal een programma zonder REM-statements sneller werken dan met.

Inleidingen zijn vaak niet erg interessant. Daarom zal ik het hierbij laten, zodat we met het echte werk kunnen beginnen. Ik wens u daarbij vele leerzame uurtjes, die een aantal voor u nuttige toepassingen zullen opleveren, of die u op het spoor van een aantal leuke toepassingen zullen zetten.

INHOUD

1. CONVERTEREN VAN GETALLEN

Een van de eerste problemen die je tegenkomt wanneer je je met computers gaat bezighouden is dat computers (intern) niet op dezelfde manier rekenen als wij mensen. De computer kent eigenlijk maar twee cijfers, de nul en de een. Met deze twee cijfers worden dan getallen weergegeven. Een gevolg van dit kleine aantal verschillende cijfers is dat getallen van wat hogere waarden met erg veel cijfers moeten worden weergegeven. Dit maakt het voor mensen erg moeilijk om deze getallen foutloos te lezen en te onthouden. Computer-vaklieden zijn daarom al lang geleden begonnen de nulletjes en eentjes in groepjes van drie of vier weer te geven. Op die manier konden de computergetallen gemakkelijk worden omgezet naar zo'n groepje van drie of vier, en de verkregen getallen konden bovendien gemakkelijk door mensen worden onthouden.

Een stelsel waarin slechts twee cijfers voorkomen, de nul en de een, heet een tweetallig of binair stelsel. Rekenen in het tweetallig stelsel wordt ook wel "binair rekenen" genoemd.
Wij mensen zijn gewend om getallen samen te stellen uit tien verschillende cijfers, de nul tot en met de negen. Wij rekenen dus in het tientallig of decimale stelsel. De volgende tabel laat in de eerste kolom het tellen in het decimale stelsel zien. In de tweede kolom ziet u de overeenkomstige waarde in het binaire stelsel. U ziet dat in het binaire stelsel, om het decimale cijfer acht weer te geven, al vier cijfers nodig zijn. De derde kolom laat zien hoe de decimale waarden uit de eerste kolom in het achttallige (octale) stelsel worden weergegeven. Daarbij valt op dat het decimale getal acht in het octale stelsel al uit twee cijfers bestaat. De vierde en laatste kolom laat zien hoe de getallen in het zestientallige (hexadecimale) stelsel worden weergegeven. Een bijzonderheid hierbij is dat wij, doordat we slechts tien verschillende cijfers kennen, zes nieuwe cijfers hebben moeten bedenken. Voor deze zes nieuwe cijfers zijn gemakshalve de eerste zes letters van het alfabet gekozen. Zo ziet u dat de waarde 10 in het hexadecimale stelsel wordt weergegeven met het "cijfer" A.

Als u in de kolom van binaire getallen allemaal nulletjes voor de getallen denkt, en u neemt dan vanaf het meest rechtse cijfer een groepje van drie cijfers, dan ziet u dat er binnen zo'n groepje steeds acht verschillende combinaties mogelijk zijn. Het octale stelsel (derde kolom) heeft precies acht verschillende cijfers. Je kunt dus een binair getal prima opdelen in groepjes van drie cijfers, en ieder groepje ver-

vangen door een octaal cijfer. Dit maakt de getallen stukken korter, terwijl het omzetten van binair naar octaal erg gemakkelijk is.

decimaal	binair	octaal	hexadecimaal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Op overeenkomstige manier zou je een binair getal ook in groepjes van vier cijfers kunnen verdelen. Met vier binaire cijfers zijn 16 verschillende combinaties mogelijk. Het hexadecimale stelsel kent zestien verschillende cijfers. Ieder groepje van vier binaire cijfers kan dus gemakkelijk worden vervangen door een hexadecimaal cijfer. Uit de tabel blijkt duidelijk dat dit voor wat betreft de lengte van de getallen aanzienlijke voordelen biedt.

Laten we nu eens kijken wat de betekenis van ieder cijfer in een getal eigenlijk is. We zullen daartoe het decimale getal 162 eens onder de loep nemen. 162 bestaat eigenlijk uit de volgende onderdelen:

$$1x100 + 6x10 + 2x1$$

Je zou deze onderdelen ook op een andere manier kunnen opschrijven:

$$1x10^2 + 6x10^1 + 2x10^0$$

U weet dat, wanneer we het getal tien opschrijven, we eigenlijk be-

doelen te zeggen tien tot de eerste macht. Voor het gemak laten we dan de macht weg. Dat tien tot de nulde macht 1 is zal misschien niet iedereen duidelijk zijn. Toch is dit vrij eenvoudig aan te tonen. Daartoe zullen we een eenvoudige deling uitvoeren:

$100:10=10$ ofwel $10^2:10^1=10^1$

U ziet dat deze deling in feite neer komt op het aftrekken van de machten ($2-1=1$). We zullen nu een andere deling op dezelfde manier uitvoeren:

$10^2:10^2=10^0$ ofwel $100:100=1$

Hieruit moge duidelijk zijn dat 10^0 gelijk is aan 1.

Uit het voorgaande is hoop ik duidelijk geworden dat de plaats van een cijfer binnen een getal een relatie heeft met de macht waartoe men het grondtal, van het stelsel waarin dat getal is geschreven, moet verheffen. Als dit waar is voor het tientallige stelsel, dan zou dat ook waar moeten zijn voor ieder ander stelsel. Laten we als voorbeeld eens een binair getal nemen. In plaats van het binaire getal 1011 zou dan ook moeten kunnen worden geschreven:

$1x2^3 + 0x2^2 + 1x2^1 + 1x2^0$

Dit maakt het omzetten naar het decimale stelsel wel erg makkelijk, er staat immers:

$1x8 + 0x4 + 1x2 + 1x1 = 11$ (decimaal)

Bovenstaande regels gaan inderdaad op voor ieder talstelsel.
Probeert u maar eens een paar getallen in de verschillende stelsels. Daarbij dient u er wel rekening mee te houden dat de cijfers die u in die getallen gebruikt nooit gelijk aan of hoger dan de aanduiding van het talstelsel mogen zijn. In het viertallige stelsel mag u dus alleen de cijfers 0, 1, 2 en 3 gebruiken, die dan worden vermenigvuldigd met de respectievelijke machten van vier.

Tot zover de theorie. Zoals ik in het begin van dit hoofdstuk al opmerkte, is een van de eerste problemen die je tegenkomt bij het werken met computers het gebruik van verschillende talstelsels. Dit maakt dat een programmeur vaak getallen uit het ene stelsel moet omzetten (converteren) naar getallen in een ander stelsel. Wanneer u zich gaat

bezighouden met machinetaalprogrammeren, dan zult u te maken krijgen met het hexadecimale stelsel, terwijl u in BASIC gewend was alles decimaal op te geven.
In MSX-BASIC bestaan een aantal functies, met behulp waarvan u getallen binair, octaal of hexadecimaal kunt ingeven. U hebt dus al gauw te maken met een aantal verschillende talstelsels.

1.1 Conversie van decimaal naar hexadecimaal

De meest voorkomende conversie is wel die van decimaal naar hexadecimaal en omgekeerd. Listing 1–1 geeft een programma waarmee getallen uit het decimale stelsel kunnen worden omgezet naar het hexadecimale stelsel. In regel 1010 wordt het te converteren decimale getal gevraagd en in variabele n opgeslagen. Regel 1020 drukt de tot dan bekende gegevens al vast af op het beeldscherm. Deze PRINT-statement wordt afgesloten met de puntkomma, zodat het eerstvolgende karakter dat wordt ge-PRINT onmiddellijk volgt op het laatst afgedrukte karakter. Daar een waarde afhankelijk van het talstelsel waarin die wordt weergegeven uit een verschillend aantal cijfers kan bestaan, wordt in regels 1030 tot en met 1050 uitgezocht uit hoeveel cijfers het geconverteerde getal zal gaan bestaan. Dit willen we weten omdat we dat later in het programma nodig hebben. U ziet dat in regel 1040 het decimale getal n wordt gedeeld door een macht van 16 (we willen immers naar het zestientallige stelsel converteren). Zodra de uitkomst van die deling kleiner dan 1 is, weten we het aantal cijfers.

Listing 1-1

```
1000 REM ************************
1001 REM * dec - hex conversie *
1002 REM ************************
1003 '
1005 CLS
1010 INPUT "Decimaal getal";N
1020 PRINT "Dec. ";N;" = Hex. ";
1030 FOR A=1 TO 100
1040 IF INT (N/(16^A))=0 THEN LET I=A:GOTO 1060
1050 NEXT A
1059 '
1060 REM *** conversie ***
```

```
1061 '
1070 FOR A=I TO 1 STEP -1
1080 LET H=INT (N/16^(A-1))
1090 LET N=N-16^(A-1)*H
1100 IF H>=10 THEN LET H=H+55
1110 IF H<10 THEN LET H=H+48
1120 PRINT CHR$ (H);
1130 NEXT A
1140 PRINT
1150 GOTO 1010
```

Voorbeeld 1-1

```
Dec.  0      = Hex. 0
Dec.  1      = Hex. 1
Dec.  10     = Hex. A
Dec.  100    = Hex. 64
Dec.  1000   = Hex. 3E8
Dec.  16383  = Hex. 3FFF
Dec.  16384  = Hex. 4000
Dec.  65535  = Hex. FFFF
Dec.  65536  = Hex. 10000
Dec.  1234567  = Hex. 12D687
```

Waarschijnlijk is een voorbeeldje het duidelijkst. Stel dat we het decimale getal 20 hadden ingegeven. De eerste keer dat we regel 1040 uitvoeren geldt dat a=1. Hierdoor zal het resultaat van de deling 20 gedeeld door 16 tot de macht 1 groter dan 1 zijn. We voeren de lus nogmaals uit, doch nu geldt dat a=2. De deling wordt dan ook 20 gedeeld door 16 tot de macht 2. Het resultaat van deze deling is kleiner dan 1. De functie INT zorgt er voor dat alle cijfers achter de komma worden weggelaten, en dus is het resultaat nu 0. Op dit moment geven we variabele i de waarde van variabele a. Deze waarde is 2, ofwel het hexadecimale getal zal uit twee cijfers gaan bestaan.

We weten nu dat we een getal van i hexadecimale cijfers moeten gaan maken. In regel 1070 starten we daarom een FOR NEXT lus die we i maal zullen doorlopen. We beginnen met de grootst mogelijke macht van 16 die nog van het decimale getal kan worden afgetrokken. Regel 1080 bepaalt het eerste (meest linkse) cijfer van het hexadecimale getal. Het hexadecimale getal n wordt gedeeld door 16 tot de macht (a−1). Als we het voorbeeld van zoeven nog eens aanhalen, dan zou dit betekenen dat we 20 delen door 16 tot de macht (2−1)=1. 16 tot

de macht 1 is 16, dus de deling is 20 gedeeld door 16. Dit is iets meer dan 1. De INT-functie zorgt er echter voor dat de cijfers achter de komma worden weggelaten, zodat het eerste hexadecimale cijfer een 1 is. Nu trekken we in regel 1090 de waarde van het eerste cijfer af van het hexadecimale getal, en we zetten het verschil in de variabele n.

In variabele h hebben we nu een numerieke waarde staan. Deze waarde kan varieren tussen 0 en 15. Zoals eerder in dit hoofdstuk opgemerkt, zouden we alle cijfers boven de 9 weergeven met een letter (A t/m F). Het is echter onmogelijk om letters in numerieke variabelen op te slaan. Vandaar dat we met regels 1100 en 1110 de numerieke waarden omzetten naar een ASCII-code. Als de computer een letter A naar het beeldscherm moet sturen, dan gebruikt hij daarvoor de code (decimaal) 65. Als hij een cijfer 0 naar het beeldscherm wil sturen dan gebruikt hij de code 48. Om nu de numerieke waarde uit variabele h om te zetten naar een bijbehorende ASCII-code zullen we iets bij die numerieke waarde op moeten tellen. Wanneer die waarde kleiner is dan 10, moeten we er 48 bijtellen. Bij een waarde van 10 tot en met 15 moeten we 55 optellen. Hierna kunnen we de numerieke variabele h gebruiken om er een karakter mee te PRINTen. Dit ziet u in regel 1120. Indien nog niet alle hexadecimale cijfers zijn afgedrukt, dan zal de lus nogmaals worden doorlopen. Zijn alle cijfers afgedrukt, dan wordt teruggesprongen naar regel 1010, waar u een nieuw decimaal getal kunt ingeven, dat vervolgens weer naar hexadecimaal wordt geconverteerd. U ziet in het voorbeeld dat dit programma in staat is om heel grote getallen te converteren. De onnauwkeurigheid van de computer bij erg grote getallen gebiedt mij echter u te waarschuwen dat er bij die grote getallen zo nu en dan een klein afrondingsfoutje kan optreden. Later in dit hoofdstuk zal ik hier nog op terugkomen.

Behalve de zojuist beschreven methode van converteren is er nog een andere manier. Daarbij wordt dan gebruik gemaakt van een zogenaamde "hex-tabel". Zoals we in het eerste programma zagen, moesten we een bepaalde waarde, afhankelijk van de waarde in variabele "h", bijtellen. We kunnen echter de waarde in variabele h ook gebruiken als een wijzer naar een karakter uit de hex-tabel. Dus, als in variabele h de waarde 0 staat, dan moeten we het eerste karakter uit de hex-tabel afdrukken, staat in h de waarde 15, dan moeten we het laatste karakter uit de tabel afdrukken.

Listing 1-2

```
1000 REM ***********************
1001 REM * Dec - Hex conversie *
1002 REM *    met HEX-tabel    *
1003 REM ***********************
1004 '
1010 LET H$="0123456789ABCDEF"
1020 LET R$=""
1030 INPUT "decimaal getal";N
1040 PRINT "Dec. ";N;" = Hex. ";
1050 FOR A=1 TO 100
1060 IF INT(N/(16^A))=0 THEN LET I=A:GOTO 1080
1070 NEXT A
1079 '
1080 REM *** conversie ***
1081 '
1090 FOR A=I TO 1 STEP -1
1100 LET H=INT(N/16^(A-1))
1110 LET N=N-16^(A-1)*H
1120 LET R$=R$+MID$(H$,(H+1),1)
1130 NEXT A
1140 PRINT R$
1150 GOTO 1020
```

Voorbeeld 1-2

```
Dec.  0  = Hex. 0
Dec.  1  = Hex. 1
Dec.  10  = Hex. A
Dec.  100  = Hex. 64
Dec.  1000  = Hex. 3E8
Dec.  1023  = Hex. 3FF
Dec.  1024  = Hex. 400
Dec.  65535  = Hex. FFFF
Dec.  65536  = Hex. 10000
Dec.  1234567  = Hex. 12D687
```

In regel 1010 van het programma uit listing 1–2 ziet u de "hex-tabel". In regel 1020 wordt variabele r$ leeg gemaakt, omdat we de verkregen cijfers daar straks in zullen gaan zetten. Regels 1050 tot en met 1070 dienen weer om het aantal cijfers in het geconverteerde resul-

taat te bepalen. Regels 1100 en 1110 bepalen de waarden van de cijfers. Het verschil met het vorige programma zit hem in de regel 1120. Hierin zeggen we:

> Laat r$ gelijk worden aan de oude inhoud van r$ met daaraan vastgeplakt het (h+1)de karakter uit de hex-tabel.

Dus indien r$ nog leeg was, dan zal r$ indien in h de waarde 3 stond, nu worden: "3". Indien bij de volgende doorloop h de waarde 11 zou bevatten, dan zou r$ "3B" worden, etc.
In regel 1140 wordt r$ afgedrukt op het beeldscherm, direct achter de tekst die daar al was afgedrukt met regel 1040. Het resultaat ziet u in het voorbeeld van de output van het programma. Met opzet heb ik daar dezelfde getallen laten converteren als in het voorbeeld bij het vorige programma, zodat u duidelijk kunt zien dat beide programma's hetzelfde resultaat geven. Dit resultaat wordt alleen op een iets andere manier bereikt.

1.2 Conversie van decimaal naar binair

Ook het converteren van decimale naar binaire getallen zult u vaak moeten doen. Listing 1–3 geeft een programma dat deze conversie uitvoert weer. In regel 1010 wordt weer naar het te converteren getal gevraagd, waarna dat getal in regel 1020 wordt afgedrukt met een =teken er achter. De reden om hier geen verdere tekst aan toe te voegen is dat het binaire getal wel eens uit erg veel cijfers (nulletjes en eentjes) kan bestaan. Om dan toch nog alle cijfertjes van dat getal op dezelfde regel te houden moet zover mogelijk aan het begin van de regel worden begonnen met afdrukken. Regels 1030 tot en met 1050 dienen weer om te bepalen uit hoeveel cijfers het geconverteerde getal zal gaan bestaan. Daarna worden de cijfers bepaald in de regels 1080 tot en met 1110.

Listing 1-3

```
1000 REM ************************
1001 REM * dec - bin conversie *
1002 REM ************************
1003 '
1010 INPUT "Decimaal getal";N
1020 PRINT STR$(N);" =";
```

```
1030 FOR A=1 TO 100
1040 IF INT (N/(2^A))=0 THEN LET I=A:GOTO 1060
1050 NEXT A
1059 '
1060 REM *** conversie ***
1061 '
1070 FOR A=I TO 1 STEP -1
1073 '
1074 REM ***********************
1075 REM * 0.4 ter compensatie *
1076 REM * onnauwkeurigheid in *
1077 REM * machtsverheffen     *
1078 REM ***********************
1079 '
1080 LET B=INT ((N+.4)/2^(A-1))
1090 LET N=N-2^(A-1)*B
1100 PRINT STR$(B);
1110 NEXT A
1120 PRINT
1130 GOTO 1010
```

voorbeeld 1-3

```
 0 = 0
 1 = 1
 10 = 1 0 1 0
 12 = 1 1 0 0
 15 = 1 1 1 1
 16 = 1 0 0 0 0
 123 = 1 1 1 1 0 1 1
 1023 = 1 1 1 1 1 1 1 1 1 1
 1024 = 1 0 0 0 0 0 0 0 0 0 0
 16000 = 1 1 1 1 1 0 1 0 0 0 0 0 0 0
```

Nu doet er zich een vreemde situatie voor. Bij het testen van dit programmaatje bleek het resultaat bij het converteren van grote getallen soms een foutje te bevatten. De oorzaak hiervan is de onnauwkeurigheid in het rekenen van de computer. Vat u dit niet als al te negatief op, want diezelfde onnauwkeurigheid komt ook voor op veel duurdere professionele 8-bits microcomputers. Bij normale berekeningen zou deze onnauwkeurigheid ook te verwaarlozen klein zijn, doch in regel 1080 wordt aan variabele b alleen het gehele deel van het resultaat van de daarachter staande berekening toegekend. Stel nu

eens dat het resultaat 0,99999999 zou zijn. Onder normale omstandigheden zou je zeggen dat dat gelijk aan 1 is. Indien je echter van dat getal alleen het gehele deel neemt, dus het deel dat voor de komma staat, dan krijg je 0. En dit is wel een groot verschil met de gewenste 1.

De hiervoor omschreven onnauwkeurigheid is er de oorzaak van dat ik in regel 1080 een waarde 0,4 heb opgeteld bij variabele n. Indien zich nu de situatie zou voordoen dat n/2 tot de macht (a–1) 0,99999999 zou zijn, dan is het vrij zeker dat het resultaat van n+0,4/2 tot de macht a–1 iets meer zal zijn dan 1. Daar met INT alleen het gehele deel van het resultaat aan b wordt toegekend zal b de waarde 1 krijgen, precies zoals we wensten. Bovendien is 0,4 niet zo groot dat het er ooit voor zal kunnen zorgen dat de waarde die aan b gaat worden toegekend per ongeluk een 2 wordt in plaats van de gewenste 1 of een 1 in plaats van de gewenste 0.

1.3 Conversie van decimaal naar een willekeurig grondtal

Wanneer u de programma's uit de vorige listings goed bekijkt, dan zal het u opvallen dat ze als twee druppels water op elkaar lijken. Het zou dus eigenlijk niet zo moeilijk moeten zijn om een programma te schrijven dat getallen kan converteren vanuit het decimale stelsel naar ieder gewenst ander talstelsel. Dat dit inderdaad geen kunststuk is bewijst het programma uit listing 1–4:

Listing 1-4

```
1000 REM ************************
1001 REM * dec - grondtal conv *
1002 REM ************************
1003 '
1005 CLS
1010 LET H$="0123456789ABCDEF"
1020 INPUT "Decimaal getal";N
1030 INPUT "Grondtal";G
1040 IF G<2 OR G>16 THEN 1030
1050 PRINT "Dec.";N;"= ";
1060 FOR A=0 TO 100
1070 IF INT (N/G^A)=0 THEN LET I=A:GOTO 1090
1080 NEXT A
1089 '
1090 REM *** conversie ***
1091 '
```

```
1100 FOR A=I TO 1 STEP -1
1110 LET R=INT((N+.4)/G^(A-1))
1120 LET N=N-(G^(A-1))*R
1130 PRINT MID$(H$,(R+1),1);
1140 NEXT A
1150 PRINT
1160 PRINT "in het";G;"-tallig stelsel."
1170 PRINT
1180 GOTO 1020
```

```
Voorbeeld 1-4
Dec. 65535 = 1111111111111111
in het 2 -tallig stelsel.

Dec. 65535 = FFFF
in het 16 -tallig stelsel.

Dec. 65535 = 65535
in het 10 -tallig stelsel.

Dec. 65535 = 177777
in het 8 -tallig stelsel.

Dec. 65535 = 10022220020
in het 3 -tallig stelsel.
```

U ziet dat daar weer gebruik is gemaakt van de methode met de hex-tabel. Het feit dat in deze tabel slechts de cijfers 0 tot en met F staan houdt meteen in dat getallen alleen geconverteerd kunnen worden naar de stelsels met de grondtallen 2 tot en met 16. Mocht u dat echter niet voldoende vinden, dan is er niets op tegen om de hex-tabel uit te breiden met een aantal cijfers. Er zijn per slot van rekening nog 20 letters in het alfabet waaraan we ook een cijferwaarde zouden kunnen toekennen. Als u dit inderdaad doet, dan moet u ook regel 1040 aanpassen. Die regel zorgt er namelijk voor dat er geen grondtal kan worden gekozen dat hogere cijfers heeft dan er in de hex-tabel voorkomen.

De variabele g, waar u in regel 1030 het gewenste grondtal in hebt gezet, wordt nu in regel 1070 gebruikt bij het bepalen van het aantal cijfers in het geconverteerde getal. Vervolgens wordt g weer gebruikt in de formules in de regels 1110 en 1120. Indien u zich variabele g even voorstelt als de waarde 2, dan ziet u dat dit programmadeel exact gelijk is aan de overeenkomstige regels uit het programma van listing 1–3. Behoudens een paar kleine wijzigingen in de PRINT-output

is het programma ook niet veel anders dan de hiervoor beschreven programma's. Het resultaat is echter wel veel universeler geworden, hetgeen u kunt zien aan de bijgevoegde voorbeelden.

1.4 Conversie van hexadecimaal naar decimaal

Nu we in staat zijn om decimale getallen naar ieder ander gewenst grondtal om te zetten, zou het ook wel prettig zijn om getallen vanuit een ander grondtal om te zetten naar decimaal. Hiertoe zullen we eerst van een praktisch voorbeeld uitgaan, het converteren van een hexadecimaal getal naar een decimaal getal.
De problemen die we hierbij kunnen tegenkomen zijn weer heel anders dan in de vorige programma's. Listing 1–5 laat het programma dat de conversie uitvoert zien.

Listing 1-5

```
1000 REM ************************
1001 REM * hex - dec conversie *
1002 REM ************************
1003 '
1005 CLS
1010 LET C$="0123456789ABCDEF"
1020 INPUT "Te converteren hex. getal";H$
1030 LET DEC=0
1040 LET L=LEN(H$)
1050 FOR A=1 TO L
1055 Z$=MID$(H$,A,1)
1060 IF Z$<"a" OR Z$>"f" GOTO 1070
1065 MID$(H$,A,1)=CHR$(ASC(Z$)-32)
1070 IF Z$<"0" OR Z$>"f" GOTO 1020
1080 NEXT A
1090 FOR A=1 TO L
1100 FOR B=1 TO 16
1110 IF MID$(C$,B,1)=MID$(H$,A,1) THEN DEC=
DEC+(B-1)*16^(L-A)
1120 NEXT B
1130 NEXT A
1140 PRINT "Hex. ";H$;" = dec.";DEC
1145 PRINT
1150 GOTO 1020
```

```
Voorbeeld 1-5

Hex. 0 = dec. 0

Hex. 1 = dec. 1

Hex. A = dec. 10

Hex. A = dec. 10

Hex. FA00 = dec. 64000

Hex. FFFF = dec. 65535
```

In regel 1010 is weer een hex-tabel gedefinieerd. Hier heb ik de hex-tabel in variabele c$ gezet, omdat het te converteren hexadecimale getal in h$ wordt gezet. Regel 1030 initieert de variabele "dec" en kent daar de waarde 0 aan toe. Deze variabele zullen we gebruiken om de geconverteerde decimale waarde in op te slaan. Regel 1040 kent het aantal karakters in de ingegeven hexadecimale waarde toe aan de variabele len. Nu volgt een belangrijk stukje. Regels 1050 tot en met 1080 controleren namelijk of de ingegeven hexadecimale waarde wel werkelijk hexadecimaal is. Bovendien wordt ervoor gezorgd dat, indien u in plaats van een A per ongeluk een a zou hebben ingegeven, die a wordt omgezet naar A. Dit wordt natuurlijk voor alle hexadecimale lettercijfers gedaan. Het gevolg hiervan is dat u vrij bent om hoofdletters en kleine letters door elkaar te gebruiken. Indien in het hexadecimale getal echter een cijfer voorkomt dat kleiner dan 0 of groter dan F is, dan wordt dat getal niet geaccepteerd, en zult u met regel 1020 worden gevraagd een nieuw getal in te geven. Zodra een correct hexadecimaal getal is ingegeven, wordt met de regels 1090 tot en met 1130 dat hexadecimale getal omgezet naar een decimale waarde, die in variabele dec wordt opgeslagen. In h$, de variabele waarin u het hexadecimale getal hebt ingegeven, staan de ASCII-codes van de door u ingetoetste cijfers. Door nu deze ASCII-code op te zoeken in de hex-tabel c$, en de plaats binnen c$ waar die ASCII-code wordt gevonden in b te onthouden, wordt de decimale waarde van het hexadecimale cijfer gevonden.
Even een klein voorbeeldje ter verduidelijking:

Stel dat het hexadecimale karakter in h$ de letter D is. Zodra we de FOR-NEXT-lus van regel 1100 binnenkomen, wordt gekeken of het eerste karakter uit c$ gelijk is aan de letter in h$ (b is bij het binnen komen van de lus 1). Daar beide karakters niet gelijk aan elkaar zijn,

wordt dus gekeken of het karakter in h$ gelijk is aan het tweede karakter uit c$. Dit gaat net zo lang door tot beide karakters aan elkaar gelijk zijn. Pas dan wordt de LET-instructie op regel 1110 uitgevoerd. Nu is D het veertiende karakter uit c$, zodat op het moment van gelijkheid de inhoud van variabele b gelijk zal zijn aan veertien. Daar D echter de waarde 13 vertegenwoordigt, moet er 1 van b worden afgetrokken voordat we er mee kunnen rekenen. De inhoud van variabele dec wordt opgeteld bij het resultaat van de berekening aan het einde van regel 1110. Indien in ons voorbeeld het gevonden karakter het enige karakter in h$ was (a is dan dus 1, evenals L), dan zal het resultaat van de berekening zijn:

dec+(14–1=13)*16 tot de macht (1–1=0)

Daar 16 tot de macht 0 gelijk is aan 1 mag je dus ook lezen:

dec+13*1=dec+13

En aangezien dec in regel 1030 de waarde nul had gekregen zal de nieuwe waarde van dec dus 13 zijn. U kunt dit resultaat nog even controleren met behulp van de aan het begin van dit hoofdstuk gegeven conversie tabel. Indien u echter begrepen hebt hoe de conversie werkt, dan neem ik aan dat u dat niet nodig zult vinden. Hebt u dit niet goed begrepen, dan zou ik u willen aanraden het begin van dit hoofdstuk nog eens aandachtig door te lezen.

1.5 Conversie van een willekeurig grondtal naar decimaal

In principe kan met het programma uit de vorige paragraaf ook ieder ander talstelsel naar het decimale stelsel worden geconverteerd. Listing 1–6 laat dit zien. De controle op de correctheid van het opgegeven te converteren getal is hier iets anders gerealiseerd dan in het vorige programma. Regel 1060 controleert of de cijfers uit n$ wel kleiner zijn dan het door u opgegeven grondtal. Indien u had opgegeven een getal uit het drietallig stelsel te willen converteren, dan wordt met regel 1060 gekeken of de door u in n$ gebruikte cijfers groter zijn dan c$(3). c$(3) is het cijfer 2. Indien een cijfer wordt gevonden dat groter is dan 2, dan moet u met regel 1040 een nieuw getal ingeven.

```
1000 REM ************************
1001 REM * grondtal - dec conv *
1002 REM ************************
1003 '
1010 LET C$="0123456789ABCDEF"
1020 INPUT "Grondtal ";G
1030 IF G<2 OR G>16 THEN GOTO 1020
1040 INPUT "Te converteren getal ";N$
1050 FOR A=1 TO LEN (N$)
1060 IF MID$(N$,A,1)>MID$(C$,G,1) GOTO 1040
1070 NEXT A
1080 LET L=LEN (N$)
1090 LET D=0
1100 FOR A=1 TO L
1110 FOR B=1 TO 16
1120 IF MID$(C$,B,1)=MID$(N$,A,1) THEN D=
D+(B-1)*G^(L-A)
1130 NEXT B
1140 NEXT A
1150 PRINT "Waarde ";N$;" in grondtal ";G
1160 PRINT "Geeft decimaal: ";D
1170 PRINT
1180 GOTO 1020
```

Voorbeeld 1-6

```
Waarde 9999 in grondtal  10
Geeft decimaal:  9999

Waarde FFFF in grondtal  16
Geeft decimaal:  65535

Waarde 1010101010 in grondtal  2
Geeft decimaal:  682

Waarde 1234567 in grondtal  8
Geeft decimaal:  342391
```

Daar de rest van dit programma geen principiële wijzigingen bevat ten opzichte van het vorige programma zal ik daar niet verder op ingaan.

1.6 Conversie van grondtal naar grondtal

Met alle tot nu toe opgedane kennis en ervaring mag het programma uit listing 1–7 geen enkel geheim meer voor u hebben. In feite doet dit programma niet meer dan we al in de vorige programma's hebben gezien. Het converteert een getal uit een willekeurig grondtal naar een decimaal getal, en aansluitend daarop wordt dat decimale getal geconverteerd naar een getal uit een ander willekeurig grondtal. Juist omdat dit programma geen nieuwe technieken bevat zal ik er hier niet verder op in gaan.
Bovendien heb ik het programma voorzien van commentaar door middel van een aantal REM-statements. U kunt bij het intikken van dit programma uiteraard die REM-statements weglaten zonder de werking van het programma te verstoren.

Listing 1-7

```
1000 REM ***********************
1001 REM * Conversie van getal *
1002 REM * in het ene grondtal *
1003 REM * naar ander grondtal *
1004 REM ***********************
1005 '
1006 CLS
1010 K$="0123456789ABCDEF"
1020 INPUT "Van welk grondtal";G1
1030 IF G1<2 OR G1>16 GOTO 1020
1040 INPUT "Naar welk grondtal";G2
1050 IF G2<2 OR G2>16 GOTO 1040
1060 INPUT "Te converteren getal";N$
1064 '
1065 REM ***********************
1066 REM * geldigheidscontrole *
1067 REM ***********************
1068 '
1070 FOR A=1 TO LEN(N$)
1080 IF MID$(N$,A,1)>MID$(K$,G1,1) GOTO 1060
1090 NEXT A
1094 '
1095 REM ***********************
1096 REM * conv. naar decimaal *
1097 REM ***********************
```

```
1098 '
1100 D=0
1110 FOR A=1 TO LEN(N$)
1120 FOR B=1 TO 16
1130 IF MID$(K$,B,1)<>MID$(N$,A,1) GOTO 1140
1135 D=D+(B-1)*G1^(LEN(N$)-A)
1140 NEXT B
1150 NEXT A
1154 '
1155 REM ***********************
1156 REM * Bep. aantal cijfers *
1157 REM ***********************
1158 '
1160 FOR A=0 TO 100
1170 IF INT(D/G2^A)=0 THEN I=A:GOTO 1200
1180 NEXT A
1190 '
1192 REM ***********************
1194 REM * Conv. naar grondtal *
1196 REM ***********************
1198 '
1200 G$=""
1210 D1=D
1220 FOR A=I TO 1 STEP -1
1230 G=INT(D1/G2^(A-1))
1240 D1=D1-G*G2^(A-1)
1250 G$=G$+MID$(K$,(G+1),1)
1260 NEXT A
1264 '
1265 REM ***********************
1266 REM * Afdrukken resultaat *
1267 REM ***********************
1268 '
1270 PRINT "De waarde ";N$
1280 PRINT "in het";G1;"-tallig stelsel"
1290 PRINT "geeft de waarde ";G$
1300 PRINT "in het";G2;"-tallig stelsel"
1310 PRINT "De decimale waarde is";D
1320 PRINT
1330 GOTO 1020
```

Voorbeeld 1-7

```
De waarde 111111111111
```

```
in het 2 -tallig stelsel
geeft de waarde FFF
in het 16 -tallig stelsel
De decimale waarde is 4095

De waarde FFFF
in het 16 -tallig stelsel
geeft de waarde 177777
in het 8 -tallig stelsel
De decimale waarde is 65535

De waarde 12345
in het 6 -tallig stelsel
geeft de waarde 5303
in het 7 -tallig stelsel
De decimale waarde is 1865
```

1.7 MSX-conversiefuncties

In MSX-BASIC hebt u de beschikking over een aantal conversiefuncties; BIN$ (dec. getal), HEX$ (dec. getal) en OCT$ (dec. getal). Hiermee kunt u decimale getallen omzetten naar respectievelijk het binaire, hexadecimale en octale talstelsel.

BIN$(7) zal als resultaat de string "111" hebben.
HEX$(15) zal als resultaat de string "F" hebben.

In een programma kunt u deze functies als volgt gebruiken:

```
1000 INPUT "Decimaal getal"; d
1010 PRINT HEX$(d)
1020 GOTO 1000
```

Het in te geven decimale getal mag nooit groter dan 32767 of kleiner dan –32768 zijn.

2. PRIEMGETALLEN

Als voorbereiding op het ontbinden in factoren van getallen, zullen we in dit hoofdstuk eens wat nader ingaan op die getallen die uit slechts 1 factor bestaan, dat wil zeggen, getallen die door geen enkel ander (lager) getal deelbaar zijn zonder een rest over te laten, de zogenaamde priemgetallen.
Het getal 2 bijvoorbeeld kan alleen door 2 worden gedeeld, waarna de uitkomst van de deling 1 is en de rest 0. Dit is ook het geval met het getal 3. Dit is wel te delen door twee, maar er blijft een rest van 1 over. We zeggen dan dat twee geen factor van het getal drie is. Deel je drie door drie, dan is het resultaat weer 1, en de rest 0. Met het getal 4 is dit heel anders. Deel je dat door 2, dan is het resultaat van die deling 2, en dat is weer deelbaar door 2. We zeggen dan dat het getal 4 uit twee factoren bestaat: 2 en 2.

2.1 Het principe van het bepalen van priemgetallen

Zolang het om kleine getallen gaat is het nog gemakkelijk voor ons, mensen, om te bepalen of ze alleen door zichzelf of ook nog door andere (kleinere) getallen deelbaar zijn. Zodra de getallen echter wat groter worden, gaat het ons erg veel tijd kosten om te zien of ze "priemgetallen" zijn. We zouden dan zoveel delingen moeten uitvoeren, dat we voor het bepalen van het al of niet priem zijn misschien dagen nodig zouden hebben. Zelfs voor een computer betekent dat een grote hoeveelheid rekenwerk. Rekentijden van enkele minuten of zelfs uren zijn dan niet buitengewoon.

Er is echter een manier om het aantal delingen drastisch te beperken. We hoeven namelijk alleen maar te delen door alle getallen tot en met de wortel uit het getal waarvan we willen weten of het een priemgetal is. Dus, het getal 9 hoeven we alleen te delen door alle getallen van 2 tot en met 3. Immers de wortel uit 9 is 3. Dit principe is toegepast in het programma uit listing 2–1:

```
Listing 2-1
1000 REM ************************
1001 REM *   Priemgetallen 1    *
1002 REM ************************
1003 '
1010 PRINT "Van welk getal wilt u weten of het"
```

```
1012 PRINT "of het een priemgetal is"
1014 INPUT G
1020 FOR A=2 TO SQR (G)
1030 IF G/A=INT(G/A) GOTO 1070
1040 NEXT A
1050 PRINT G;" is een priemgetal."
1060 GOTO 1010
1070 PRINT TAB(4);G;" is geen priemgetal."
1075 PRINT
1180 GOTO 1010
```

Voorbeeld 2-1

```
     123  is geen priemgetal.
     234  is geen priemgetal.
     345  is geen priemgetal.
 4567  is een priemgetal.
     5678  is geen priemgetal.
 7  is een priemgetal.
     147  is geen priemgetal.
     987  is geen priemgetal.
     873  is geen priemgetal.
     543  is geen priemgetal.
 431  is een priemgetal.
```

In dat programma wordt op regel 1010 gevraagd naar het getal waarvan moet worden bepaald of het een priemgetal is. Vervolgens wordt in de regels 1020 tot en met 1040 uitgezocht of het betreffende getal een priemgetal is. Regel 1020 bepaalt de delers. De kleinste deler zal 2 zijn, de grootste de wortel uit het opgegeven getal. Met regel 1030 wordt gekeken of het gegeven getal deelbaar is door de deler zonder een rest over te laten. Is zo'n deling inderdaad mogelijk, dan is het gegeven getal blijkbaar geen priemgetal, en wordt naar regel 1070 gesprongen.
Is zo'n deling niet mogelijk dan moet de deler worden verhoogd, en wordt regel 1030 nog eens uitgevoerd. Als alle getallen tot en met de wortel uit het gegeven getal zijn geprobeerd zonder dat de deling mogelijk was, dan is het gegeven getal een priemgetal en wordt naar regel 1050 gesprongen.

Bij een test gaf ik het getal 123457 in, en na 7 seconden wist de computer dat dit een priemgetal was. Daarna heb ik regel 1020 als volgt gewijzigd:

```
1020 FOR a=2 TO g–1
```

Hiermee worden dus alle waarden van 2 tot het gevraagde getal gedeeld op dat gevraagde getal. Weer gaf ik het getal 123457 op, doch na enkele minuten heb ik het programma onderbroken. Toen bleek dat er nog maar ruim 5000 delingen waren uitgevoerd. De wortel uit 123457 is ruim 351. Door die truc met de wortel worden dus aanzienlijke tijdsbesparingen bereikt.

2.2 Een lijst van priemgetallen genereren

Op basis van het in het vorige programma gegeven principe moet het mogelijk zijn de computer gewoon alle priemgetallen te laten afdrukken. Hiertoe hoeven we er alleen maar zorg voor te dragen dat we de computer steeds een getal opgeven en laten bepalen of het een priemgetal is. Is het een priemgetal dan laten we dat afdrukken. Na ieder getal, priem of niet priem herhalen we de procedure nadat we het vorige getal hebben verhoogd met 1.

```
1000 REM ********************
1001 REM * Priemgetallen 2 *
1002 REM ********************
1003 '
1010 LET G=1
1020 LET G=G+1
1030 LET A=2
1040 IF SQR (G)<A THEN PRINT G:GOTO 1020
1050 IF G/A=INT(G/A) GOTO 1020
1060 A=A+1
1070 GOTO 1040
```

Voorbeeld 2-2

```
 2
 3
 5
 7
 11
 13
 17
 19
 23
 29
 31
 37
 41
```

Het programma uit listing 2–2 maakt zo'n lijst van priemgetallen. In plaats van een FOR NEXT lus is nu gekozen voor het gebruik van een GOTO-instructie. Bij het begin zal in "g" de waarde 2 staan. Daarna wordt 2 in "a" gezet. Indien de wortel uit "g" kleiner is dan de waarde in "a" dan is het getal in "g" een priemgetal. Nu wordt "g" met 1 verhoogd en is dus 3. Weer wordt in "a" een 2 gezet en blijkt dat de wortel uit 3 kleiner is dan 2, en dus dat 3 een priemgetal is. Weer wordt "g" met 1 verhoogd. Nu staat er 4 in "g". De wortel uit 4 is 2. Dat is niet minder dan de waarde in "a" en dus wordt nu de volgende regel (1050) uitgevoerd. Daarin blijkt dat 4 gedeeld door 2 geen rest heeft, en dus dat 4 geen priemgetal is. Daarom wordt "g" weer met 1 verhoogd.

Door het programma op deze manier met een wat groter getal door te lopen wordt de functie van variabele "a" duidelijk. Eerst zal worden gekeken of het getal deelbaar is door 2. Zoniet, dan wordt "a" verhoogd met 1 en wordt gekeken of het getal deelbaar is door 3, etc.

2.3 Vlugger genereren van priemgetallenlijst

Het vorige programma kan erg lang bezig zijn met het genereren van priemgetallen. Bij metingen bleek dat het 1 minuut en 37 seconden duurde om alle priemgetallen tot en met 199 te bepalen. En dat zijn nog maar 45 priemgetallen. Het zou dus prettig zijn als het mogelijk was om dat programma nog wat te versnellen.

Dit is inderdaad mogelijk. Het programma uit listing 2–3 geeft aan hoe dat in zijn werk gaat. Als eenmaal vastgesteld is dat 2 een priemgetal is, dan kunnen we alle even getallen die groter zijn dan 2 buiten beschouwing laten. Met andere woorden we kijken niet meer of 8 een priemgetal is, want 8 is deelbaar door twee. We kijken echter wel naar 9, ook al blijkt dat achteraf ook geen priemgetal te zijn. Door alleen de oneven getallen op priem zijn te onderzoeken zou dus een besparing van 50 procent moeten worden bereikt.

Listing 2-3

```
1000 REM **********************
1001 REM *   Priemgetallen 3 *
1002 REM **********************
```

```
1003 '
1010 LET G=1
1020 PRINT 2
1030 LET G=G+2
1040 LET A=3
1050 IF SQR(G)<A THEN PRINT G:GOTO 1030
1060 IF G/A=INT(G/A) GOTO 1030
1070 LET A=A+2
1080 GOTO 1050
```

Voorbeeld 2-3

```
 3
 5
 7
 11
 13
 17
 19
 23
 29
 31
 37
 41
 43
 47
 53
 59
 61
```

Uit regels 1010 en 1030 blijkt dat we beginnen met het onderzoeken van het getal 3. Het volgende getal zal 3+2 zijn, het volgende 5+2, etc. Daar oneven getallen nooit deelbaar zijn door even getallen, heeft het ook geen zin om de deler ooit even te laten worden. Vandaar dat in regel 1070 die deler steeds met twee wordt verhoogd. Voor het overige werkt dit programma precies als het vorige. Bij meting bleek echter dat het bepalen van de eerste 45 priemgetallen slechts 43 seconden kostte, hetgeen dus inderdaad al een stuk sneller is dan het vorige programma. Op de zojuist beschreven manier zou het ook nog mogelijk zijn om alle getallen die deelbaar zijn door drie weg te laten, doch de besparing in tijd zal dan al aanzienlijk minder spectaculair zijn. Vandaar dat er hier niet verder op zal worden ingegegaan.

2.4 Priemgetallen in een tabel

De snelste manier om te bepalen of een getal een priemgetal is of niet, is om alle priemgetallen die er zijn in een tabel te zetten. Je kunt dan, om te bepalen of een bepaald getal een priemgetal is, eenvoudigweg die tabel raadplegen. Het programma uit listing 2–4 laat een manier zien hoe zo'n tabel kan worden gegenereerd en geraadpleegd.

Listing 2-4

```
1000 REM *********************
1010 REM *   Priemgetallen   *
1020 REM *********************
1025 '
1026 CLS
1030 DIM P(64)
1040 LET G=1
1050 LET X=1
1060 LET P(X)=2
1070 LET X=X+1
1080 IF X>64 GOTO 1160
1090 LET G=G+2
1100 LET A=3
1110 IF SQR (G)<A THEN P(X)=G:GOTO 1070
1120 IF G/A=INT(G/A) GOTO 1090
1130 LET A=A+2
1140 GOTO 1110
1150 '
1151 REM ************************
1152 REM * Afdrukken priemtabel *
1153 REM ************************
1154 '
1160 PRINT "Priem-tabel:"
1170 FOR Y=1 TO 8
1175 FOR X=1 TO 8
1180 LOCATE X*4,Y+4
1190 PRINT P(X+(Y-1)*8);
1200 NEXT X
1210 NEXT Y
1220 PRINT
1230 PRINT "Van welk getal tussen 1 en 312"
```

```
1233 PRINT "wilt u weten of het priem is?"
1236 INPUT N
1240 IF N<2 OR N>312 GOTO 1230
1250 FOR A=1 TO 64
1260 IF N=P(A) THEN PRINT N;" is priem":GOTO 1230
1270 NEXT A
1280 PRINT N;" is niet priem"
1290 GOTO 1230
```

Voorbeeld 2-4

```
Priem-tabel:



      2   3   5   7   11  13  17  19
      23  29  31  37  41  43  47  53
      59  61  67  71  73  79  83  89
      97  101 103 107 109 113 127 131
      137 139 149 151 157 163 167 173
      179 181 191 193 197 199 211 223
      227 229 233 239 241 251 257 263
      269 271 277 281 283 293 307 311

Van welk getal tussen 1 en 312
wilt u weten of het priem is?
? 159
 159  is niet priem
```

De tabel is een numerieke array "p", die uit 64 velden bestaat. In regel 1030 wordt die tabel geïnitieerd. Er is nu een extra tellertje nodig om het gegenereerde priemgetal in het volgende veld van de tabel te kunnen zetten. Deze teller heeft hier de naam "x" gekregen. In regel 1080 wordt gecontroleerd of er al 64 velden zijn gevuld. Indien dit inderdaad zo is, dan wordt naar regel 1160 gesprongen. Hier wordt de zojuist gegenereerde tabel op het beeldscherm afgedrukt. Daarbij worden steeds 8 kolommen per regel gedrukt.

Vanaf regel 1230 wordt de mogelijkheid gegeven om van een bepaald getal te laten uitzoeken of het een priemgetal is. Daar in de tabel slechts priemgetallen tot en met 311 staan, kunnen uit deze tabel alleen getallen tot en met 311 worden gezocht. Het eigenlijke zoeken

wordt in regel 1260 gedaan. Daar worden alle velden uit de array één voor één gelezen en vergeleken met het in regel 1230 ingegeven getal. Daarna wordt op het scherm afgedrukt of het opgegeven getal al of niet in de tabel voorkomt.

Mocht u het aantal priemgetallen in deze tabel wel erg klein vinden, dan is hier eenvoudig verbetering in te brengen. Het aantal gewenste priemgetallen kunt u op regels 1030, 1080 en 1250 in plaats van het daar gegeven aantal (64) intypen.
In regel 1170 kunt u het aantal af te drukken regels (met ieder 8 priemgetallen) opgeven.
Daarna moet regel 1240 nog worden aangepast, opdat er ook grotere getallen kunnen worden opgevraagd. Hoe groter de tabel wordt gemaakt, hoe langer het zal duren om hem te maken.

3. ONTBINDEN EN VEREENVOUDIGEN

Zoals we in het vorige hoofdstuk hebben gezien zijn priemgetallen die getallen die uit slechts één factor bestaan. De meeste getallen bestaan echter uit meer dan één factor. Zo bestaat het getal 6 uit de factoren 2 en 3. Door beide factoren met elkaar te vermenigvuldigen ontstaat weer het getal 6. Hoe groter het getal, hoe moeilijker het wordt om te bepalen uit welke factor of factoren dat getal bestaat. We zullen in dit hoofdstuk eens zien hoe we de computer getallen voor ons kunnen laten ontbinden.

3.1 Ontbinden in factoren

Van een gegeven getal kun je bepalen uit welke factoren dat getal bestaat door dat getal te delen door alle getallen die kleiner zijn dat het gegeven getal. Stel dat het getal 6 is, dan zul je eerst moeten proberen of 6 te delen is door 2. Als dit inderdaad mogelijk is, en dat is het, dan is 2 een factor van 6. Daarop moet dan de deling worden uitgevoerd, waarna er nog 3 overblijft. Nu moet je kijken of drie door 2 te delen is. Dit is niet mogelijk, tenminste niet zonder een rest over te houden. Daarom verhogen we 2 met 1, en proberen of 3 deelbaar is door 3 (2+1). Dit is inderdaad mogelijk, dus is 3 ook een factor van het getal 6. Weer voeren we de deling uit, doch nu blijkt het resultaat 1 te zijn. Hierdoor weten we dat het getal 6 uit niet meer factoren bestaat.

Listing 3-1

```
1000 REM ***************************
1001 REM * Ontbinden van factoren *
1002 REM ***************************
1010 INPUT "Welk getal mag ik voor u ontbinden ";N
1020 PRINT N;" = ";
1030 LET A=2
1040 IF N/A=INT(N/A) THEN PRINT A;" ";:N=N/A:GOTO 1040
1050 A=A+1
1060 IF A<=N GOTO 1040
1070 PRINT
1080 GOTO 1010
```

```
Voorbeeld 3-1

 4    =  2    2
 8    =  2    2    2
 12   =  2    2    3
 125  =  5    5    5
 200  =  2    2    2    5    5
 3456 =  2    2    2    2    2    2    2    3    3    3
 4567 =  4567
 321  =  3    107
```

Dit principe is ook verwerkt in het programma van listing 3–1. In regel 1030 zetten we de waarde 2 (de kleinst mogelijke factor) in variabele "a". Daarna kijken we in regel 1040 of er bij deling van het in regel 1010 opgegeven getal door de factor in "a" een rest overblijft. Indien er geen rest overblijft drukken we "a" af. Daarna delen we het orginele getal door "a" en kijken nog eens of het nieuwe getal deelbaar is door de factor in "a". Dit blijft net zo lang doorgaan totdat blijkt dat de deling een rest oplevert. Op dat moment tellen we (met regel 1050) 1 op bij de factor in "a". Zolang "a"niet groter is dan het nog te ontbinden deel van het opgegeven getal gaan we hierna weer verder met het onderzoek in regel 1040. Het gevolg van dit programma ziet u in het voorbeeld bij het programma.

3.2 Vereenvoudigen van breuken

Het vereenvoudigen van breuken heeft alles te maken met ontbinden in factoren. In feite zou je zowel teller als noemer van de te vereenvoudigen breuk eerst moeten ontbinden, om ze vervolgens beide door de overeenkomstige factoren te delen.
Er is echter een eenvoudiger manier.

Een voorbeeldje zal waarschijnlijk het beste zijn om deze eenvoudige manier van vereenvoudigen van breuken te demonstreren. Met teller en noemer van de breuk voeren we domweg de deling uit, en controleren daarna of de rest van die deling 0 is. Is de rest 0, dan is de noemer, waarbij die rest 0 werd, het getal waardoor zowel teller als noemer deelbaar zijn. Is de rest niet 0, dan wordt een nieuwe deling uitgevoerd, doch nu wordt de vorige noemer gedeeld door de rest.
Het schema ziet er als volgt uit:

teller : noemer := rest
↓ ↓
teller : noemer := rest
↓ ↓
teller : noemer := rest

rest = 0? dan is de noemer van dat moment het getal waardoor de orginele teller en noemer kunnen worden gedeeld.
Dit voorbeeld nog eens maar nu met de breuk 18/30:

18 : 30 = 0, rest = 18

30 : 18 = 1, rest = 12

18 : 12 = 1, rest = 6

12 : 6 = 2, rest = 0

Hieruit blijkt dat zowel de orginele teller (18) als de orginele noemer (30) te delen zijn door 6. Hiermee wordt de vereenvoudigde breuk: 3/5.

Voorgaand principe is gebruikt in het programma van listing 3–2. In regel 1060 wordt de deling uitgevoerd en wordt gekeken of de rest 0 is. Indien dit zo is, wordt naar regel 1110 gesprongen. Daar wordt de teller vereenvoudigd. Daarna wordt in regel 1120 de noemer vereenvoudigd. De rest van het programma dient om de verkregen resultaten op een nette manier op het beeldscherm af te drukken. Een voorbeeld daarvan is bij het programma gegeven.

Listing 3-2

```
1000 REM ************************
1001 REM *  vereenvoudigen van  *
1002 REM *  breuken             *
1003 REM ************************
1004 '
1005 CLS
1010 PRINT "Geeft u svp de teller en de noemer"
1015 PRINT "van de door u te bepalen breuk."
1020 INPUT "TELLER";T
1030 INPUT "NOEMER";N
1040 LET T1=T
```

```
1050 LET N1=N
1060 LET R=T1-INT (T1/N1)*N1
1070 IF R=0 THEN 1110
1080 LET T1=N1
1090 LET N1=R
1100 GOTO 1060
1110 LET T1=T/N1
1120 LET N1=N/N1
1130 LET S$="-------------"
1140 LOCATE 0,5:PRINT "teller ";T
1145 LOCATE 22,5:PRINT T1
1150 LET S1=LEN (STR$(T))
1160 LET S2=LEN (STR$(T1))
1165 LOCATE 7,6:PRINT LEFT$(S$,S1)
1170 LOCATE 16,6:PRINT "wordt"
1175 LOCATE 22,6:PRINT LEFT$(S$,S2)
1180 LOCATE 0,7:PRINT "noemer ";N
1185 LOCATE 22,7:PRINT N1
1190 LOCATE 0,20:INPUT "Volgende breuk (j/n)";A$
1200 IF A$<>"n" AND A$<>"j" GOTO 1190
1210 IF A$="n" THEN STOP
1220 CLS
1230 GOTO 1010
```

Voorbeeld 3-2

```
Geeft u svp de teller en de noemer
van de door u te bepalen breuk.
TELLER? 58198140
NOEMER? 44618574

teller  58198140        30
        ---------wordt ---
noemer  44618574        23
```

Het gegeven voorbeeld werd in een fractie van een seconde door de computer gemaakt.

Na al het gegoochel met getallen uit de vorige hoofdstukken, zullen we nu, om toch nog een beetje in de sfeer te blijven, die getallen eens op een rijtje zetten. Verschillende, voor allerlei toepassingen interessante technieken zullen daarbij aan de orde komen.

Bij een groot aantal spellen is het mogelijk dat er meerdere spelers kunnen deelnemen. Aan het eind van dat spel is dan degene die de meeste (of minste) punten heeft vergaard de winnaar. Om te bepalen wie dat is moet dan uit alle resultaten de hoogste of de laagste score worden gezocht. Dit zoeken naar het hoogste getal uit een rij is natuurlijk niet alleen belangrijk bij spelprogramma's. Er zijn ook legio serieuze toepassingen te bedenken waarin dat moet.

Behalve het zoeken van een hoogste of laagste waarde uit een rij, kan het ook zijn dat er gezocht moet worden naar een bepaalde waarde. Indien de rij erg lang is, zal het zoeken naar die bepaalde waarde, afhankelijk van de plaats binnen die rij, erg lang kunnen duren. Door nu die rij te sorteren kan een speciale (snelle) zoekmethode worden toegepast. Het sorteren zelf zal echter aanzienlijk langer duren dan het eenmalig zoeken in een ongesorteerde rij. Sorteren heeft dan ook pas zin wanneer er meer dan eens in dezelfde rij naar een getal moet worden gezocht. Zowel een aantal verschillende sorteermethodes als de speciale zoekmethode zullen in dit hoofdstuk worden besproken en van voorbeeldprogramma's worden voorzien.

4.1 Zoek het grootste getal uit een rij

Om een getal uit een rij te kunnen zoeken zullen we eerst een rij getallen moeten hebben. In alle voorbeeldprogramma's in dit hoofdstuk is daartoe gebruik gemaakt van een array die wordt gevuld met willekeurige getallen. Zo ook in het programma uit listing 4–1.
In regel 1010 ziet u dat het programma vraagt hoeveel van die willekeurige getallen moeten worden gegenereerd.
Vervolgens wordt de array gedimensioneerd in regel 1020. Met de regels 1030 tot en met 1050 wordt het gewenste aantal willekeurige getallen gemaakt en in de array geplaatst. Met de functie RND wordt een getal tussen 0 en 1 met 14 cijfers achter de komma gegenereerd. Door dat getal met 1000 te vermenigvuldigen ontstaat een waarde die tussen 0 en bijna 1000 ligt. Door van dat getal alleen het gehele deel (het deel voor de komma) te nemen met de functie INT, wordt een

getal dat kan varieren van 0 tot en met 999 in de array gezet.

```
1000 REM **********************
1001 REM * Generen van de ry *
1002 REM **********************
1003 '
1010 INPUT "Hoeveel getallen wenst u";N
1020 DIM G(N)
1030 FOR A=1 TO N
1040 LET G(A)=INT(RND(-TIME)*1000)
1050 NEXT A
1059 '
1060 REM *************************
1061 REM * Zoeken grootste getal *
1062 REM *************************
1063 '
1070 LET GROOTSTE=G(1)
1080 FOR A=2 TO N
1090 IF GROOTSTE<G(A) THEN LET GROOTSTE=G(A)
1100 NEXT A
1109 '
1110 REM ***********************
1111 REM * Afdrukken resultaat *
1112 REM ***********************
1113 '
1120 PRINT "Van de getallen:"
1130 FOR A=1 TO N STEP 2
1140 PRINT G(A),G(A+1)
1150 NEXT
1160 PRINT "is ";GROOTSTE;" de grootste."
1170 END
```

Voorbeeld 4-1

```
Van de getallen:
 355            760
 165            165
 571            976
 381            786
 191            191
 596            2
 407            812
 217            217
 622            28
```

```
 433              838
 243              648
 648              54
 459              864
 269              674
 674              80
is  976  de grootste.
```

Stel nu dat we een array met 20 velden hebben gemaakt, dan is de kans groot dat er 20 verschillende waarden in die array komen te staan. Het is wel zeker dat die waarden kris kras door elkaar staan. Het is dus zeker niet zo dat je kunt zeggen dat het getal 999, als dat al voor komt, aan het einde van de tabel zal staan. Dit maakt het dan ook noodzakelijk om, bij het zoeken naar de hoogste waarde, altijd de hele array door te lopen. Steeds wanneer een volgende waarde hoger blijkt te zijn dan de vorige, dan zal de vorige waarde moeten worden vergeten en de nieuwe onthouden. Pas wanneer alle velden van de array zijn onderzocht kan worden gezegd wat de hoogste waarde is.

Volgens dit principe werkt het programmadeel van regel 1060 tot en met 1100. We beginnen met het definieren van een variabele, die ik voor het gemak de naam "grootste" heb gegeven. Hierin zetten we het eerste getal uit de array. We gaan er, zolang we nog niet beter weten, van uit dat dat eerste veld de hoogste waarde heeft. Nu lezen we met de regels 1080 tot en met 1100 steeds een volgend veld, dus alle velden vanaf het tweede tot het einde van de array (van 2 TO n). Zodra het getal in variabele "grootste" kleiner blijkt te zijn dan het op enig moment onderzochte veld van de array, dan wordt die hogere waarde uit de array in grootste gezet (LET grootste=g(a)). Vanaf dat moment is er dus een nieuwe hoogste waarde. Dit gaat zo door tot het einde van de array is bereikt. Op dat moment bevat de variabele "grootste" werkelijk de hoogste waarde.

Daar wij natuurlijk niet geloven dat de computer zo snel kan bepalen wat de hoogste waarde is, willen we controleren of het correct is gedaan. Daartoe dienen de regels 1120 tot en met het einde. Daar worden eerst alle getallen, precies zoals ze in de array voorkomen afgedrukt, vervolgens wordt het hoogste getal aangegeven.

4.2 Zoek het kleinste getal uit een rij

Het principe van dit programma (zie listing 4–2) is precies gelijk aan het vorige. Nu heb ik echter de variabele "grootste" een voor de-

ze toepassing meer geschikte naam gegeven, namelijk "kleinste". Het belangrijkste verschil zit hem echter in regel 1090. Hierin wordt gevraagd of "kleinste" groter is dan een getal uit de array. Is dit zo dan wordt die waarde uit de array aan "kleinste" toegekend. In feite komt het er dus op neer dat alleen het 'kleiner dan teken' nu is vervangen door een 'groter dan teken'. De grote overeenkomst met het vorige programma maakt verdere uitleg overbodig. Later in dit boek zult u nog een programma tegenkomen waarin dit principe van het zoeken van de grootste of kleinste waarde wordt toegepast.

Listing 4-2

```
1000 REM ***********************
1001 REM * Genereren van de ry *
1002 REM ***********************
1003 '
1010 INPUT "Hoeveel getallen wenst u";N
1020 DIM G(N)
1030 FOR A=1 TO N
1040 LET G(A)=INT (RND(-TIME)*1000)
1050 NEXT
1059 '
1060 REM *************************
1061 REM * Zoeken kleinste getal *
1062 REM *************************
1063 '
1070 LET KLEINSTE=G(1)
1080 FOR A=2 TO N
1090 IF KLEINSTE>G(A) THEN LET KLEINSTE=G(A)
1100 NEXT
1109 '
1110 REM ***********************
1111 REM * Afdrukken resultaat *
1112 REM ***********************
1113 '
1120 PRINT "Van de getallen:"
1130 FOR A=1 TO N STEP 2
1140 PRINT G(A),G(A+1)
1150 NEXT
1160 PRINT "is ";KLEINSTE;" de kleinste."
1170 END
```

```
Voorbeeld 4-2

Van de getallen:
 933            338
 743            149
 554            959
 959            364
 769            175
 580            580
 985            390
 795            201
is  149  de kleinste.
```

4.3 Sorteren met zoeken van de hoogste waarde

Indien u met een van de hiervoor gegeven programma's een grote array heeft aangemaakt, dan zal het u zijn opgevallen dat er nogal wat tijd voor nodig is om de hoogste of laagste waarde te bepalen. Dit zou ook het geval zijn indien u niet naar de hoogste waarde, maar zo maar naar een bepaalde waarde zou vragen. Zeker wanneer dan blijkt dat die gevraagde waarde niet in de array voorkomt. In dat geval moet namelijk net als bij het bepalen van de hoogste waarde de hele array worden doorgezocht.
Wanneer er in dezelfde tabel meer dan eens moet worden gezocht, dan verdient het aanbeveling om die tabel te sorteren. Er is namelijk een goede en snelle manier om te onderzoeken of een bepaalde waarde wel of niet in een tabel voor komt, de binaire zoekmethode.

Ons eerste probleem is dus het sorteren van een ongesorteerde array. Dit is echter niet zo moeilijk als het lijkt. We zijn al in staat om het hoogste getal uit een array te zoeken. Stel nu eens dat we een array van 4 getallen hebben. Als we nu van de eerste drie getallen uitzoeken welke de hoogste is, dan vergelijken we die met het vierde getal. Is dat vierde getal lager dan wisselen we het hoogste getal om met het vierde. We weten nu absoluut zeker dat het hoogste getal op de vierde plaats staat. Nu zoeken we het hoogste getal uit de eerste twee plaatsen van de array, en we vergelijken dat met het derde getal uit de array. Weer zullen we het grootste getal, indien nodig, van plaats verwisselen, nu echter met het derde getal.
Nu zijn er nog maar twee getallen over en het enige dat we nog moeten doen is die twee met elkaar vergelijken, en indien nodig van plaats laten verwisselen.

Listing 4-3

```
1000 CLS
1005 INPUT "Hoeveel getallen wenst u in de array";N
1010 LET M=N
1020 DIM O(N):REM ongesorteerd
1030 DIM S(N):REM gesorteerd
1040 FOR A=1 TO N
1050 LET O(A)=INT (RND(-TIME)*1000)
1060 LET S(A)=O(A)
1070 NEXT A
1079 '
1080 REM ************************
1081 REM * Eerst grootste getal *
1082 REM * zoeken. Plaats daar- *
1083 REM * van onthouden        *
1084 REM ************************
1085 '
1090 LET GROOTSTE=S(1)
1100 LET INDEX=1
1110 FOR A=2 TO N-1
1120 IF GROOTSTE<S(A) THEN GROOTSTE=S(A):LET INDEX=A
1130 NEXT A
1139 '
1140 REM ************************
1141 REM * Indien grootste gro- *
1142 REM * ter dan laatste, dan *
1143 REM * omwisselen           *
1144 REM ************************
1145 '
1150 IF GROOTSTE>S(N) THEN S(INDEX)=S(N):S(N)=GROOTSTE
1160 LET N=N-1
1170 IF N>=2 GOTO 1090
1179 '
1180 REM ***********************
1181 REM * Afdrukken resultaat *
1182 REM ***********************
1183 '
1190 PRINT "ongesorteerd","gesorteerd"
1200 FOR A=1 TO M
1210 PRINT O(A),S(A)
1220 NEXT A
1230 END
```

```
Voorbeeld 4-3

ongesorteerd  gesorteerd
 812           27
 217           53
 622           217
 27            243
 432           432
 243           458
 648           622
 53            648
 458           812
 863           863
```

Het uitzoeken van wat het grootste getal is wordt in het programma van listing 4–3 gedaan met de regels 1090 tot en met 1130. Dit is praktisch een copie van het programma uit paragraaf 4.1. Het enige verschil zit in regel 1110, waarmee nu alleen de velden tot en met het een na laatste (n–1) worden bekeken.
Verder wordt in regel 1120 de plaats van het grootste getal binnen de array in variabele "index" onthouden. Op het moment dat we regel 1150 beginnen uit te voeren hebben we dus twee gegevens, de hoogste waarde in "grootste" en de plaats van die waarde in "index".

In regel 1150 kijken we dan of de gevonden hoogste waarde hoger is dan de waarde van het laatste veld uit de array. Zo ja, dan wordt het laatste veld naar de plaats binnen de array waar de hoogste waarde stond gecopieerd, en wordt de waarde uit de variabele "grootste" naar het laatste veld in de array geschreven. Op dit moment staat dus het hoogste getal in het laatste veld van de array. We kunnen nu rustig stellen dat we dat laatste veld nooit meer hoeven te controleren, en laten dat veld buiten verdere beschouwing door variabele "n" met 1 te verlagen (regel 1160). Om nu te voorkomen dat we eeuwig door zouden gaan met sorteren, wordt in regel 1170 gekeken of we alle velden gesorteerd hebben, en zo niet dan wordt teruggesprongen naar regel 1090.

In de regels 1050 en 1060 hebben we de willekeurige getallen steeds in twee arrays gezet, array "o" en array "s". Vervolgens hebben we alleen array "s" gesorteerd. We kunnen daardoor na het sorteren zowel de ongesorteerde als de gesorteerde array afdrukken. Dit is meer voor de ongelovigen of nieuwsgierigen onder ons gedaan. Het afdrukken wordt in de regels 1190 tot en met 1220 gedaan. Na listing 4–3 ziet u het resultaat hiervan in het bijgevoegde voorbeeld.

4.4 De binaire zoekmethode

We zijn nu in staat om een gesorteerde array te produceren. Het eerste deel van het programma uit listing 4–4 is in feite een copie van het programma uit listing 4–3. Vanaf regel 1190 echter komt er iets geheel nieuws. Dat laatste deel van het programma zal de gesorteerde array volgens de binaire zoekmethode controleren op het voorkomen van een door u te bepalen waarde.
Laten we maar eens gaan kijken wat er precies gebeurt.

In het eerste veld van de gesorteerde array staat de laagste waarde en in het laatste veld de hoogste waarde. Zoals al eerder opgemerkt kunnen we nu op een wat elegantere manier zoeken naar het voorkomen van een bepaalde waarde in die array. De binaire zoekmethode werkt als volgt. Om er achter te komen of een bepaalde waarde voorkomt in de rij, lees je eerst het middelste getal uit die rij. Dat vergelijk je dan met het gezochte getal. Er zijn nu drie mogelijkheden; het gezochte getal is gevonden, het gezochte getal is hoger dan het gelezen getal, of het gezochte getal is lager. In het eerste geval is het programma succesvol beëindigd. Indien het gezochte getal hoger is dan het gelezen getal, dan moet het tussen het gelezen getal en het einde van de array zitten. We lezen dus het getal dat daar midden tussen in zit, en vergelijken dit weer met het gezochte getal. Weer zijn er de al genoemde drie mogelijkheden. Dit gaat zo door tot het getal is gevonden, of tot het midden gelijk wordt aan boven en ondergrens. In dat laatste geval weten we dat het gezochte getal niet in de array voorkomt. Het volgende voorbeeld laat zien wat er gebeurt.
Het getal dat we zoeken is 6.

```
Rij:  1    2    4    5    6    9    12   20   21   23   34
      ----------------------------------------------------
                                1
      --------------------------
                      2
                     -----------
                           3
```

In de eerste zoekbeurt wordt het midden van de hele rij gelezen. Daar staat de waarde 9. De gezochte waarde is lager, dus gaan we in de tweede zoekbeurt het midden van het linker deel van de rij lezen, hier vinden we de waarde 5. Daar 6 hoger is dan 5 moeten we nu het midden van het deel van de rij dat rechts ligt van de laatste gelezen waarde en links van het midden van de hele rij lezen. In die derde

zoekbeurt vinden we inderdaad het gezochte getal. Op deze manier kan, vooral bij wat grotere arrays een geweldige tijdsbesparing worden verkregen. In vier zoekbeurten kunnen al 16 velden worden onderzocht. In vijf beurten al 32. Dit loopt op met machten van 2. Om een rij van 65536 velden te onderzoeken zijn slechts 16 zoekbeurten nodig.

Gewapend met de zojuist verstrekte informatie, is het lezen van het programmadeel dat volgens de binaire zoekmethode werkt (zie listing 4–4) niet moeilijk meer. Na het opvragen van het te zoeken getal in regel 1190, zetten we het nummer van het eerste veld van de array in de variabele "E", en het nummer van het laatste veld van diezelfde array in de variabele "L". Vervolgens berekenen we in regel 1220 het nummer van het middelste veld van de array en zetten dat nummer in variabele "MI".

Listing 4-4

```
1000 INPUT "Hoeveel getallen wenst u in de array";N
1010 LET M=N: X$=""
1020 DIM O(N):REM ongesorteerd
1030 DIM S(N):REM gesorteerd
1040 FOR A=1 TO N
1050 LET O(A)=INT(RND(1)*100)
1060 LET S(A)=O(A)
1070 NEXT
1079 '
1080 REM ************************
1081 REM * Eerst grootste getal *
1082 REM * zoeken. Plaats daar- *
1083 REM * onthouden.           *
1084 REM ************************
1085 '
1090 LET G=S(1)
1100 LET I=1
1110 FOR A=2 TO N-1
1120 IF G<S(A) THEN G=S(A):I=A
1130 NEXT A
1139 '
1140 REM ************************
1141 REM * Indien grootste gro- *
1142 REM * ter dan laatste, dan *
```

```
1143 REM * omwisselen.          *
1144 REM ************************
1145 '
1150 IF G>S(N) THEN S(I)=S(N):S(N)=G
1160 LET N=N-1
1170 IF N>=2 THEN GOTO 1090
1179 '
1180 REM ************************
1181 REM * binair zoeken in s() *
1182 REM ************************
1183 '
1190 PRINT "Welk getal zoekt u?";
1192 I$=INKEY$
1193 IF I$=CHR$(13) GOTO 1196
1194 IF I$<"0" OR I$>"9" GOTO 1192
1195 X$=X$+I$:GOTO 1192
1196 X=VAL(X$): X$="":PRINT X;
1200 E=1
1210 L=M
1220 MI=INT((E+L)/2)
1230 IF X=S(MI) THEN PRINT"is";MI;"E GETAL":GOTO 1190
1240 IF E=L THEN PRINT "komt niet voor.":GOTO 1190
1250 IF X<S(MI) THEN L=MI:GOTO 1220
1260 LET E=MI+1
1270 GOTO 1220
```

Voorbeeld 4-4

```
Welk getal zoekt u? 1 komt niet voor.
Welk getal zoekt u? 2 is 1 E GETAL
Welk getal zoekt u? 3 is 2 E GETAL
Welk getal zoekt u? 4 komt niet voor.
Welk getal zoekt u? 5 komt niet voor.
Welk getal zoekt u? 6 komt niet voor.
Welk getal zoekt u? 7 komt niet voor.
Welk getal zoekt u? 8 komt niet voor.
Welk getal zoekt u? 9 komt niet voor.
Welk getal zoekt u? 10 is 3 E GETAL
Welk getal zoekt u? 11 komt niet voor.
Welk getal zoekt u? 12 is 4 E GETAL
Welk getal zoekt u? 13 is 5 E GETAL
Welk getal zoekt u? 14 komt niet voor.
Welk getal zoekt u?
```

Nu lezen we het middelste veld en kijken of de inhoud daarvan gelijk is aan de gezochte waarde. Zo ja, dan is het programma beëindigd. Indien de waarde nog niet gevonden is, kijken we met regel 1240 of variabele "E" gelijk is aan "L". Zou dit namelijk zo zijn, dan wil dat zeggen dat we de hele array doorzocht hebben, zonder dat de gezochte waarde werd gevonden. Die waarde komt dus blijkbaar niet in de array voor. Zijn "E" en "L" niet gelijk aan elkaar, dan kijken we of de gezochte waarde kleiner is dan de gelezen middelste waarde. Indien dit zo is, dan betekent dit dat de gezochte waarde links van het midden moet staan. Daarom laten we variabele "E" zoals hij was, doch veranderen we variabele "L" door er de waarde van variabele "M" in te zetten (regel 1250). Hierna springen we terug naar regel 1220, waar het nieuwe midden wordt bepaald.

Zoals gezegd, het voordeel van deze zoekmethode wordt groter naarmate de te doorzoeken array groter wordt. Probeert u het maar eens uit met verschillende grootten van array "s".

4.5 Sorteren volgens de bubble up methode

Van de vele tientallen of misschien wel honderden sorteermethoden wil ik nog enkele voorbeelden geven. Iedere methode heeft zijn voor en nadelen. De ene methode is zuinig met het geheugen, doch werkt langzaam, een andere gebruikt veel geheugenruimte doch werkt snel, en weer een andere is ingewikkeld om te schrijven maar heeft daarbij voordelen voor wat betreft geheugengebruik en snelheid. Verder is de ene sorteermethode meer geschikt voor kleine hoeveelheden te sorteren getallen, terwijl de andere methode weer meer geschikt is voor grote hoeveelheden te sorteren getallen. Van alle in het verleden uitgedachte methoden is de "bubble up" methode wel één van de meest populaire. Waarschijnlijk door zijn eenvoud, want de sorteersnelheid is niet erg hoog.

Listing 4-5

```
1000 REM ***********************
1001 REM * sorteren bubble up *
1002 REM ***********************
1003 '
1010 INPUT "Hoeveel getallen wenst u ";N
1020 DIM S(N)
```

```
1030 PRINT "Array S wordt nu gevuld."
1040 FOR A=1 TO N
1050 LET S(A)=INT(RND(-TIME)*1000)
1055 FOR I=1 TO S(A)/50:NEXT I
1060 NEXT A
1070 PRINT "Sorteren wordt nu gestart."
1080 FOR A=N TO 2 STEP-1
1090 FOR B=1 TO A-1
1100 IF S(A)>S(B) GOTO 1140
1110 LET H=S(A)
1120 LET S(A)=S(B)
1130 LET S(B)=H
1140 NEXT B
1150 NEXT A
1159 '
1160 REM ************************
1161 REM * afdrukken resultaat *
1162 REM ************************
1163 '
1170 FOR A=1 TO N
1180 PRINT S(A)
1190 NEXT
```

Voorbeeld 4-5

```
Array S wordt nu gevuld.
Sorteren wordt nu gestart.
 4
 18
 56
 69
 173
 225
 329
 433
 485
 602
 640
 744
 758
 848
 862
```

Het programma uit listing 4–5 werkt volgens de bubble up methode. Na het genereren van de array in de regels 1010 tot en met 1060, wordt het sorteren in regel 1080 gestart. De hele sorteerroutine bestaat uit slechts 8 regels. Verdere uitleg lijkt mij niet nodig, met een stukje ruitjespapier kunt u eventueel zelf nagaan wat er precies gebeurt. Wel wil ik nog even opmerken dat de PRINT-instructies zijn opgenomen om u in staat te stellen de tijd die nodig is voor het genereren van de array en voor het sorteren daarvan te meten.

4.6 De binaire sorteermethode

Het laatste programma in dit hoofdstuk is een sorteerprogramma dat volgens de binaire sorteermethode werkt. Zoals u in listing 4–6 kunt zien is dit programma (het sorteerdeel tenminste) veel ingewikkelder dan de bubble up methode. Gezien uw kennis van de binaire zoekmethode, en het feit dat ik in de listing variabelenamen heb gebruikt die verklarend werken, zal ik ook dit programma niet verder bespreken.

Listing 4-6

```
1000 REM ************************
1001 REM *   binair sorteren    *
1002 REM ************************
1003 '
1010 INPUT "Hoeveel getallen wenst u ";N
1020 DIM S(N)
1030 PRINT "Array 'S' wordt nu gevuld."
1040 FOR A=1 TO N
1050 LET S(A)=INT(RND(1)*1000)
1060 NEXT A
1065 GOSUB 1300
1070 PRINT "Sorteren wordt nu gestart."
1080 FOR A=2 TO N
1090 LET LINKS=1
1100 LET RECHTS=A
1110 LET MIDDEN=INT((RECHTS-LINKS)/2)+LINKS
1120 IF S(A)<S(MIDDEN) GOTO 1150
1130 IF S(A)>S(MIDDEN) GOTO 1180
1140 LET RECHTS=MIDDEN
1150 IF RECHTS=MIDDEN GOTO 1210
1160 LET RECHTS=MIDDEN
1170 GOTO 1110
```

```
1180 IF LINKS=MIDDEN GOTO 1210
1190 LET LINKS=MIDDEN
1200 GOTO 1110
1210 LET HULP=S(A)
1220 FOR B=A TO RECHTS+1 STEP-1
1230 LET S(B)=S(B-1)
1240 NEXT B
1250 LET S(RECHTS)=HULP
1260 NEXT A
1270 GOSUB 1300
1280 END
1289 '
1290 REM ***********************
1291 REM * subrotine voor het *
1292 REM *afdrukken der arrays*
1293 REM ***********************
1294 '
1300 FOR A=1 TO N
1310 PRINT S(A)
1320 NEXT A
1330 RETURN
```

```
Voorbeeld 4-6
Array 'S' wordt nu gevuld.
 595
 106
 765
 577
 734
 184
 370
 949
Sorteren wordt nu gestart.
 106
 184
 370
 577
 595
 734
 765
 949
```

Bij het testen van dit programma bleek het sorteren van een array van 100 getallen bijna twee keer zo snel te gaan als met het bubble-up sorteerprogramma.

5. INVOER VAN GEGEVENS VIA TOETSENBORD

In de tot nu toe gegeven programma's werd meestal van de INPUT instructie gebruik gemaakt. Dit had twee redenen; ten eerste is het gemakkelijk te programmeren, ten tweede maakt de eenvoud van de INPUT-instructie de programma-listing overzichtelijk. Toch heeft de INPUT-instructie een aantal nadelen. Deze nadelen hebben mij er toe gebracht om in alle programma's die door anderen moeten worden uitgevoerd de INKEY$-functie te gebruiken. Een van de belangrijkste redenen daarvoor is, dat als je een programma waarin INPUTs staan door kinderen laat uitvoeren, zij steeds weer in staat blijken om het programma voortijdig te laten onderbreken. Ze krijgen dan een systeemboodschap, waarvan ze de betekenis niet kennen, en moeten dan het programma weer helemaal opnieuw starten.

De meest voorkomende fout is wel dat er een getal voor een numerieke variabele wordt gevraagd. Indien in plaats van een cijfer een letter wordt ingetikt, dan zal die letter door de INPUT-instructie gewoon worden geaccepteerd. Doch op het moment dat op RETURN wordt gedrukt, ziet het systeem dat er onverwerkbare gegevens zijn ingetypt. Het is dan ook het systeem dat het programma stopt en een boodschap naar het scherm stuurt. Er kleven echter nog meer nadelen aan INPUT. Vanaf het moment dat de INPUT-instructie wordt gestart, blijft de computer inderdaad op invoer van het toetsenbord wachten, en doet in de tussentijd ook niets anders meer.

Alle genoemde bezwaren van de INPUT-instructie kunnen met de INKEY$-functie worden weggenomen. Het enige bezwaar wat daar weer aan kleeft is dat het gebruik van INKEY$ meer programmeertijd kost. In de hiernavolgende paragrafen zal ik proberen aan de hand van een aantal voorbeelden de mogelijkheden van INKEY$ duidelijk te maken. Voor een juist begrip is echter enige kennis en inzicht in de ASCII-code nodig. Laten we daar dus eerst eens naar kijken.

5.1 ASCII-code

Het geheugen van de MSX computer is opgedeeld in vakjes van acht bits ieder. Een bit kan alleen nul of een zijn. De vraag is nu hoe je een letter, leesteken of cijfer hoger dan 1 in dat geheugen op kan slaan. Met acht bitjes zijn 256 combinaties te maken. Je zou dan ook kunnen besluiten om de combinatie waarin alle acht bitjes nul zijn de code voor het cijfer 0 te noemen. De combinatie 0000 0001 zou

dan het cijfer 1 kunnen voorstellen, de combinatie 0000 0010 het cijfer 2, enz. Op deze manier kun je doorgaan totdat je een code hebt bedacht voor ieder cijfer, iedere letter en ieder leesteken. Dit is een reële mogelijkheid, doch in het verleden zijn er andere codes bedacht. Een van die codes is de ASCII-code. Dit is de Amerikaanse Standaard Code voor Informatie-overdracht. Bijna alle computerfabrikanten hebben deze code als standaard aangenomen.
Het gebruik van zo'n standaard code heeft voor de computergebruiker het grote voordeel dat hij gegevens vanuit zijn computer zonder converteren naar een andere computer kan oversturen. Wie een RS 232 C Interface heeft, zal dit voordeel waarschijnlijk al aan den lijve hebben ondervonden. Doch ook degenen die bijvoorbeeld een parallel interface voor een printer aan hun computer hebben aangesloten, genieten, misschien zonder zich dit te realiseren, van de voordelen van een internationaal erkende code. De meeste printers werken namelijk ook met de ASCII-code, en kunnen zodoende de gegevens uit de computer netjes afdrukken op een stuk papier.

Voor alle duidelijkheid heb ik de ASCII-codes en de daarbij behorende karakters even op mijn printer laten afdrukken. Daarbij zal het u opvallen dat in deze lijst het £-teken ontbreekt. Mijn printer drukt in plaats daarvan een accent grave af (code 96). Voor het overige is er geen verschil.

ASCII-code tabel

32		33	!
34	"	35	#
36	$	37	%
38	&	39	'
40	(	41	)
42	*	43	+
44	,	45	-
46	.	47	/
48	0	49	1
50	2	51	3
52	4	53	5
54	6	55	7
56	8	57	9
58	:	59	;
60	<	61	=
62	>	63	?
64	@	65	A

66	B	67	C
68	D	69	E
70	F	71	G
72	H	73	I
74	J	75	K
76	L	77	M
78	N	79	O
80	P	81	Q
82	R	83	S
84	T	85	U
86	V	87	W
88	X	89	Y
90	Z	91	[
92	\	93	]
94	^	95	_
96	`	97	a
98	b	99	c
100	d	101	e
102	f	103	g
104	h	105	i
106	j	107	k
108	l	109	m
110	n	111	o
112	p	113	q
114	r	115	s
116	t	117	u
118	v	119	w
120	x	121	y
122	z	123	{
124			125	}
126			

In de tabel staan alleen de codes 32 tot en met 126. Zoals eerder al opgemerkt zijn er 256 combinaties mogelijk. Echter, ASCII maakt slechts van 7 bitjes gebruik. Het achtste en meest significante bit wordt niet gebruikt. Hierdoor zijn alle codes van 128 tot en met 256 vrij voor toekenning naar eigen smaak. MSX heeft hiervan gebruik gemaakt door daarin codes voor een aantal grafische karakters te definiëren. Wel in ASCII gedefiniëerd zijn de codes van 0 tot en met 31. De hierin gedefiniëerde codes hebben allemaal te maken met datacommunicatie en besturing. MSX heeft ook belangrijke karakters gedefiniëerd in dit gebied, zoals code 27 = ESCAPE en code 13 = RETURN. Deze codes zullen in de voorbeeldprogramma's worden

gebruikt. Verder vraag ik nog even uw aandacht voor de hoofdletters en de kleine letters. U ziet in de tabel dat de kleine letters allemaal een code hebben die een waarde 32 hoger ligt dan hun overeenkomstige hoofdletter. Ook van dit gegeven zullen we in de voorbeelden gebruik maken.

5.2 Invoer van 1 karakter

Het komt vaak voor dat je in een programma slechts een bepaald karakter wilt hebben, en indien dat bepaalde karakter niet wordt ingegeven, dan wordt het programma niet voortgezet. Zelf heb ik de gewoonte om in mijn programma's, wanneer ik de gebruiker iets laat lezen op het scherm, te wachten met verder uitvoeren totdat de gebruiker de RETURN-toets heeft ingedrukt.

Listing 5-1

```
10 GOTO 1100
1000 REM *********************
1001 REM * INKEY$ voor 1 kar *
1002 REM *********************
1003 '
1010 A$=INKEY$
1020 IF A$="" GOTO 1010
1030 RETURN
1100 CLS:PRINT "Om door te gaan dient u"
1105 PRINT "de RETURN-toets in te drukken"
1110 LOCATE 0,8
1115 PRINT "Probeer eens een andere toets"
1120 GOSUB 1010
1130 IF A$<>CHR$(13) THEN 112C
1140 CLS
1150 LOCATE 10,10:PRINT "DANK U WEL"
```

Het programma uit listing 5–1 is hiervan een voorbeeld. Het accepteren van een toets is in een subroutine ondergebracht. Dit zijn de regels 1010 tot en met 1030.

In regel 1010 wordt een eventueel ingetikt karakter in A$ geplaatst. Is er geen toets ingedrukt, dan zal A$ leeg zijn, en wordt teruggegaan naar regel 1010. Is A$ niet leeg, dan is de subroutine uitgevoerd. U ziet dus dat er in deze subroutine slechts 1 karakter wordt ingelezen. Om meerdere karakters van het toetsenbord op te halen zal de routine meerdere keren moeten worden aangeroepen.

Het programma zelf begint op regel 1100. Daar wordt u gevraagd de RETURN-toets in te drukken. Vervolgens wordt met regel 1120 de sub-routine van regel 1010 gestart en na terugkeer daarvan wordt gekeken wat er in variabele A$ staat. Is dat iets anders dan CHR$ 13 (de code voor de RETURN-toets), dan wordt teruggesprongen naar regel 1120, waarna nogmaals naar de subroutine wordt gegaan. Dit gaat zo door tot de RETURN-toets werkelijk is ingedrukt. De enige andere toets die zal worden geaccepteerd is de CONTROL + STOP. Daarmee kunt u het programma op normale manier onderbreken.

5.3 Invoer van cijfers

Het programma van listing 5–2 geeft een voorbeeld van het opvragen van cijfers. In principe werkt het op dezelfde manier als het vorige programma, alleen wordt de gebruiker nu de keuze uit een aantal verschillende toetsen gegeven. In regel 1130 ziet u dat, indien de ingetoetste waarde kleiner is dan het karakter 1, of groter dan het karakter 4, de invoer niet wordt geaccepteerd. Er wordt dan zonder iets af te drukken teruggegaan naar regel 1120, zodat de gebruiker de kans krijgt nogmaals een cijfer in te toetsen. Als u er de ASCII-code tabel op nakijkt zult u zien dat de code voor 0 lager is dan de code voor 1. De code voor vijf is hoger dan die voor vier. Regel 1130 kijkt dus zuiver naar de ASCII-codes, en op basis daarvan wordt beslist of een karakter wel of niet wordt geaccepteerd.

Listing 5-2

```
10 GOTO 1100
1000 REM ************************
1001 REM * INKEY$ voor cijfers *
1002 REM ************************
1003 '
1010 A$=INKEY$
1020 IF A$="" THEN 1010
1030 RETURN
```

```
1100 CLS:PRINT "Type een cijfer in tussen 0 en 5"
1105 LOCATE 0,5
1110 PRINT "probeer rustig andere cijfers"
1120 GOSUB 1010
1130 IF A$<"1" OR A$>"4" THEN GOTO 1120
1140 LOCATE 15,10:PRINT A$
1150 END
```

Waar in het vorige programma gebruik werd gemaakt van CHR$, wordt hier het betreffende cijfer (tussen aanhalingstekens) gebruikt. Dit kan alleen omdat dit cijfer een afdrukbaar karakter is. De RETURN-toets of de BACKSPACE-toets is dat niet. Om niet afdrukbare codes toch te kunnen gebruiken, zet men die code achter de functie CHR$.

5.4 Invoer van teksten

Als voorbeeld van de mogelijkheden van de INKEY$-functie laat het programma uit listing 5–3 zien hoeveel programmeerwerk er in een redelijk eenvoudig probleem gaat zitten. Als probleem heb ik gekozen voor het invoeren van een postcode. Die postcode bestaat uit 4 cijfers en 2 letters. Postcodes worden vaak met hoofdletters geschreven, doch er worden ook wel eens kleine letters gebruikt. Met dit programma maakt het niet uit wat je in typt, want van kleine letters worden automatisch hoofdletters gemaakt. Verder zorgt dit programma er voor dat je niet per ongeluk een letter in kan tikken terwijl er een cijfer had moeten staan en omgekeerd. Bovendien geeft dit programma een correctiemogelijkheid en laat het met behulp van een cursor zien waar het volgende karakter terecht zal komen. U ziet, het lijkt meer op een tekstverwerkingsprogramma dan op een eenvoudig invoer-voorbeeld.

Listing 5-3

```
10 LET P$="":CLS
20 GOTO 1100
1000 REM ************************
1001 REM * INKEY$ voor teksten *
1002 REM ************************
1003 '
1010 A$=INKEY$
1020 IF A$="" GOTO 1010
1030 BEEP
```

```
1040 RETURN
1059 '
1060 REM ***********************
1061 REM *    start van het     *
1062 REM *    hoofdprogramma    *
1063 REM ***********************
1064 '
1100 LOCATE 0,20
1105 PRINT "Geeft u s.v.p. uw postcode in."
1110 FOR A=1 TO 4
1115 LOCATE 12+LEN(P$),10
1120 PRINT "_"
1130 GOSUB 1010
1135 LOCATE 10,21
1140 PRINT "              "
1150 IF A$=CHR$(8) THEN P$=LEFT$(P$,LEN(P$)-1):LOC
ATE 12,10:PRINT P$;"_ ":A=A-1:GOTO 1130
1160 IF A$<"0" OR A$>"9" THEN LOCATE 10,21: PRINT
"CIJFERS SVP";:GOTO 1115
1170 LET P$=P$+A$
1180 LOCATE 12,10:PRINT P$
1190 NEXT A
1200 LET P$=P$+" "
1210 FOR A=1 TO 2
1220 LOCATE 12+LEN(P$),10:PRINT "_";
1230 GOSUB 1010
1240 LOCATE 10,21:PRINT "             ";
1250 IF A$=CHR$(8) THEN P$=LEFT$(P$,LEN(P$)-1):LOC
ATE 12,10:PRINT P$;"_ ":LET A=A-1:GOTO 1220
1260 IF A$>="a" AND A$<="z" THEN A$=CHR$(ASC(A$)-3
2)
1270 IF A$<"A" OR A$>"Z" THEN LOCATE 10,21:PRINT "
LETTERS SVP":GOTO 1220
1280 LET P$=P$+A$
1290 LOCATE 12,10:PRINT P$
1300 NEXT A
```

Regels 1010 tot en met 1040 halen de ingetoetste code op, maar geven bovendien een duidelijk hoorbare piep ten teken dat er een toets is ingedrukt. Deze subroutine wordt vanuit het hoofdprogramma dat op regel 1100 begint aangeroepen.

De regels 1110 tot en met 1190 halen de 4 cijfers voor de postcode van het toetsenbord op. Er is hier gekozen voor een constructie met

een FOR NEXT lus, doch er zou ook met een tellertje kunnen worden gewerkt. Regel 1120 zet de cursor op de plaats waar het eerste cijfer moet verschijnen. Nu wordt gekeken of er al een toets is ingedrukt met regel 1130. Met regel 1140 wordt het gebied waar de postcode moet komen te staan overschreven met spaties. Dit is gedaan om later bij het corrigeren de foute letters te laten verdwijnen. Natuurlijk verdwijnen daarmee ook de goede letters en cijfers, doch die worden bij elke doorgang weer op het scherm geschreven.

Regel 1150 kijkt of de ingetypte toets misschien de BACKSPACE-toets is. In dat geval wordt het laatste karakter van p$ weggehaald. Daarna wordt p$ opnieuw afgedrukt, gevolgd door de cursor.
Bovendien wordt a met 1 verlaagd, immers als we het derde karakter al hadden ingetikt, dan zal a de waarde 3 hebben, en bij het intikken van het volgende karakter de waarde 4. We willen echter het derde karakter nogmaals intikken. Als in regel 1150 blijkt dat de BACK-space-toets niet werd ingedrukt, dan zal worden doorgegaan met regel 1160. Daarin controleren we of het ingetypte karakter wel een cijfer is. Zo niet, dan drukken we een boodschap op het scherm af waarin we meedelen dat er alleen cijfers mogen worden ingetikt.

Was er een cijfer ingedrukt, dan wordt dat cijfer in regel 1170 toegevoegd aan p$, en met regel 1180 afgedrukt op het scherm. Hierna wordt de volgende doorgang van de FOR NEXT lus gestart en kan het volgende cijfer worden ingetikt. Zijn de vier cijfers allemaal ingetypt, dan wordt met regel 1200 een spatie afgedrukt, waarna een nieuwe FOR NEXT lus wordt gestart. In deze lus, die op dezelfde manier werkt als de vorige worden de twee letters ingevoerd. Regel 1260 kijkt of het ingetoetste karakter een kleine letter is, en zo ja dan wordt daarvan een hoofdletter gemaakt door van de ASCII-code van die kleine letter de waarde 32 af te trekken.

Misschien brengt dit programma u ertoe aan het schrijven van een tekstverwerkingsprogramma te beginnen. Daarvoor komt natuurlijk nog wel heel wat meer kijken, maar een aantal van de hier gegeven techniekjes zult u daarbij zeker kunnen gebruiken.

In dit hoofdstuk zullen een aantal onderwerpen die specifiek van toepassing zijn op MSX-computers, worden behandeld. Daarnaast zullen enkele routines aan de orde komen die gewoon leuk zijn, enerzijds omdat ze zo'n leuk resultaat geven, anderzijds omdat ze zo eenvoudig zijn.

6.1 Een digitale klok

Het programma uit listing 6–1 maakt van uw computer een digitale klok. Met regels 1010 tot en met 1080 wordt u gevraagd de juiste tijd in te geven en wordt gecontroleerd of dat wel een bestaanbare tijd is. Als de door u ingetikte tijd wordt goedgekeurd, wordt het scherm schoongemaakt met regel 1085. Nu wordt de systeemvariabele TIME op nul gezet met regel 1090. Vervolgens wordt met regel 1095 de variabele TIME weer uitgelezen. De stand van TIME wordt in variabele T1 bewaard.

Met regel 1120 wordt de systeemvariabele TIME opnieuw uitgelezen en de stand wordt opgeslagen in variabele T2. Als u nu weet dat het systeem de systeemvariabele TIME 50 keer per seconde met 1 verhoogt, dan zult u begrijpen dat het verschil tussen de inhoud van variabele T1 en T2 een goede aanwijzing is voor de verstreken tijd. Er kan zich een moeilijkheid voordoen, namelijk dat de systeemvariabele TIME op een gegeven ogenblik zijn maximale waarde heeft bereikt en als er dan nog 1 bij wordt opgeteld, zal de waarde van TIME weer 0 worden. Als dat het geval is, zal regel 1130 een negatief resultaat geven. En hoe je een negatieve tijd moet aangeven... ik weet het niet. Vandaar dat regel 1125 kijkt of de waarde van variabele T2 wel groter is dan die van T1. Zo niet, dan zal variabele T1 gelijk worden gemaakt aan T2. Dit geeft natuurlijk wel een kleine fout in de tijd, doch daar deze situatie zich slechts 1 keer in de ruim 20 minuten voordoet, is de te ontstane fout verwaarloosbaar klein.

Regel 1130 berekent het verschil tussen T2 en T1 en zet dat verschil in variabele T. Hierin staan nu dus het verstreken aantal 1/50-ste seconden. Vandaar dat, zolang het verschil kleiner is dan 50, er wordt teruggesprongen naar regel 1120, waar variabele T2 opnieuw met de inhoud van TIME wordt geladen. Zodra echter het verschil 50 of meer is, wordt met regel 1150 de variabele T1 met 50 verhoogd (1 seconde), waardoor de waarde van T1 de waarde van T2 bijna of helemaal

nadert.

Daar er nu blijkbaar een seconde is voorbijgegaan moet de klok worden bijgesteld. Regel 1160 verhoogt het aantal seconden met 1 en regel 1170 kijkt of er inmiddels meer dan 1 minuut (60 seconden) voorbij is gegaan. Is dit zo, dan zal de minutenteller met 1 worden verhoogd, en zal worden gekeken of er inmiddels een uur voorbij is gegaan (regel 1180). Tenslotte kijkt regel 1190 of er een dag voorbij is gegaan, en indien nodig wordt de urenteller op 0 gezet. Regels 1200 tot en met 1230 drukken de bijgewerkte tijd af op het beeldscherm, waarna wordt teruggekeerd naar regel 1120. Hier begint het hele verhaal opnieuw.

Deze klok is op eenvoudige wijze toe te passen in andere programma's, zoals in een andere paragraaf in dit hoofdstuk zal worden aangetoond. Het bijhouden van de tijd is vooral leuk in spelprogramma's. Het geeft dan een extra spel-element aan die programma's. Het spelletje in hoofdstuk 7 geeft hiervan een voorbeeld, al is daarin niet voor een klok gekozen.

Listing 6-1

```
1000 REM ************************
1001 REM *    digitale klok     *
1002 REM ************************
1003 '
1005 CLS
1010 INPUT "Hoe laat is het (UUMMSS)";T$
1020 IF LEN(T$)<>6 GOTO 1010
1030 U=VAL(LEFT$(T$,2))
1040 IF U>23 GOTO 1010
1050 M=VAL(MID$(T$,3,2))
1060 IF M>60 GOTO 1010
1070 S=VAL(RIGHT$(T$,2))
1080 IF S>60 GOTO 1010
1085 CLS
1090 TIME=0:LOCATE 10,M/3:PRINT "reset"
1095 T1=TIME
1100 '
1110 REM ************************
1111 REM *    tijdsbepaling     *
1112 REM ************************
```

```
1113 '
1120 T2=TIME
1125 IF T2<T1 THEN T1=T2
1130 T=T2-T1
1140 IF T<50 GOTO 1120
1150 T1=T1+50
1160 S=S+1
1170 IF S>59 THEN S=0:M=M+1
1180 IF M>59 THEN M=0:U=U+1
1190 IF U>=24 THEN U=0
1200 LOCATE 28,0
1210 PRINT USING "##";U;:PRINT ".";
1220 PRINT USING "##";M;:PRINT ".";
1230 PRINT USING "##";S
1240 GOTO 1120
```

6.2 Lichtkrant

Het nu volgende routinetje is erg eenvoudig. Het resultaat is echter zo leuk dat ik het toch graag laat zien. Met regel 1010 zet u een tekst in variabele A$. Deze tekst mag enkele letters zijn, maar mag ook meer dan een regel op het beeldscherm zijn. Wilt u erg lange teksten invoeren, dan zult u met CLEAR een grotere ruimte voor de variabelen in het geheugen moeten reserveren.

Regel 1020 zorgt er voor dat, als de tekst korter is dan 40 karakters, de tekst gecentreerd op het scherm verschijnt. Is de tekst langer dan 40 karakters, dan zullen alleen de eerste 40 karakters worden afgedrukt. Dit wordt met regel 1030 gedaan. Nadat de tekst is afgedrukt, moet de tekst in de variabele A$ worden verschoven. Het eerste karakter van de string wordt weggehaald, de rest van de string wordt naar links geschoven, en het weggehaalde karakter wordt achter aan de string geplakt. Dit alles wordt door regels 1040 en 1050 gedaan. Regel 1060 veroorzaakt een korte pauze, die nodig is omdat de tekst anders zo snel over het scherm schiet, dat het niet meer is te lezen. Hierna wordt weer teruggegaan naar regel 1020, waar de inmiddels verschoven tekst opnieuw wordt afgedrukt.

Het voorbeeld (6–2) laat de inhoud van A$ zien, steeds nadat het is afgedrukt. Op het beeldscherm komen de regels niet onder elkaar te staan, maar over elkaar heen. Hiervoor zorgt de LOCATE instructie.

Listing 6-2

```
1000 REM **********************
1001 REM *     Lichtkrant     *
1002 REM **********************
1003 '
1005 WIDTH 40: CLS
1010 INPUT "Geef svp een tekst in";A$
1015 L=LEN(A$)
1020 IF L<=40 THEN LOCATE 20-(INT(L/2)),10:PRINT A$
1030 IF L>40 THEN LOCATE 0,10: PRINT LEFT$(A$,40)
1040 B$=RIGHT$(A$,L-1)
1050 A$=B$+LEFT$(A$,1)
1060 FOR A=1 TO 50:NEXT A
1070 GOTO 1020
```

Voorbeeld 6-2

```
Dit bericht wordt door de MSX computer n
it bericht wordt door de MSX computer na
t bericht wordt door de MSX computer naa
 bericht wordt door de MSX computer naar
bericht wordt door de MSX computer naar 
ericht wordt door de MSX computer naar l
richt wordt door de MSX computer naar li
icht wordt door de MSX computer naar lin
cht wordt door de MSX computer naar link
ht wordt door de MSX computer naar links
```

6.3 Beeldscherm naar printer

Een van de dingen (en ik mag wel zeggen een van de weinige dingen) die de MSX-computer mist, is een commando om een afdruk van het beeldscherm op een printer te maken. Dit viel mij pas op toen ik dit boek aan het schrijven was. Zo nu en dan is het namelijk verduidelijkend om behalve de listing zelf ook het resultaat van het programma te laten zien. Dat resultaat staat echter op het beeldscherm. En een commando om dat beeldscherm af te drukken is er niet.

Met enige kennis van de Video RAM is het echter niet zo'n kunst om de gegevens van het beeldscherm uit dat Video RAM te lezen en naar de printer te sturen. Het Video RAM heeft een vaste indeling, die echter per mode, waarin de computer zich bevindt, anders is. Indien

de computer in tekstmode 1 is, dan geeft het register BASE(0) het startadres binnen het Video RAM waar de beeldscherminhoud begint. Is de computer in tekstmode 2, dan staat dit startadres in het register BASE(5).

Met behulp van de functie VPEEK kan het Video RAM worden uitgelezen. U ziet nu dat de routines uit listing 6–3 en 6–4 niet zo moeilijk zijn. De FOR...NEXT lussen geven het aantal uit te lezen beeldschermregels en het aantal karakters per regel aan. Vervolgens wordt het Video RAM uitgelezen en wordt de uitgelezen waarde naar de printer gestuurd met het LPRINT commando. Wilt u de routines voor uw eigen toepassing gebruiken dan zult u dus het aantal regels (24 is regel 0 t/m 23) en het aantal karakters per regel (40 karakters is 0 t/m 39) moeten invullen in de FOR...NEXT statements. Vervolgens zet u het aantal karakters per regel in regel 120 en u kunt het scherm laten afdrukken.

Listing 6-3

```
10 '*********************************
20 '* BEELDSCHERM NAAR LINE PRINTER *
30 '* IN TEKSTMODE 1 (24*40 CHARS). *
40 '*********************************
50 '
100 FOR J=0 TO 21
110 FOR I=0 TO 39
120 LPRINT CHR$(VPEEK(BASE(0)+J*40+I));
130 NEXT I
140 LPRINT
150 NEXT J
```

Listing 6-4

```
10 '*********************************
20 '* BEELDSCHERM NAAR LINE PRINTER *
30 '* IN TEKSTMODE 2 (24*32 CHARS). *
40 '*********************************
50 '
100 FOR J=0 TO 21
110 FOR I=0 TO 31
120 LPRINT CHR$(VPEEK(BASE(5)+J*32+I));
```

```
130 NEXT I
140 LPRINT
150 NEXT J
```

6.4 Sorteren met tijdsindicatie

Bij het sorteren moet vaak een bepaalde sorteermethode worden gekozen op grond van de sorteersnelheid. Die snelheid is erg afhankelijk van de gekozen sorteermethode en het aantal te sorteren items. Een gevolg is dan ook dat er nogal eens met de klok in de hand moet worden gewacht tot het sorteren klaar is. Zeker bij grote bestanden kost dat een hoop tijd. Vandaar dat ik hier een voorbeeldje heb gegeven van het inbouwen van een timer in een sorteerprogramma. Het sorteerprogramma zelf is niet nieuw. Het werd al in hoofdstuk 4 behandeld. Wat echter, behalve de klok, wel nieuw is, is dat we hier geen getallen, maar tekst sorteren. Dit sorteren van tekst gaat in dit programma zuiver op ASCII-code. Het is dus geen alfabetisch sorteren. Immers, een hoofdletter Z heeft een lagere waarde dan een kleine letter a. Hierdoor zal de Z voor de a worden gesorteerd. In hoofdstuk 12 wordt een programma gegeven dat werkelijk alfabetisch sorteert.

Daar zowel het principe van de tijdmeting als het principe van dit sorteerprogramma al bekend zijn, zal ik daar niet verder op in gaan. Hier volgt het programma en een voorbeeld van de resultaten van dat programma.

Listing 6-5

```
1000 REM ***********************
1005 REM * Tekst sorteren met *
1010 REM * tijdsindicatie.    *
1015 REM ***********************
1020 '
1030 INPUT "Hoeveel woorden wenst u in de array";N
1040 LET M=N
1050 DIM O$(N):REM ongesorteerd
1060 DIM S$(N):REM gesorteerd
1070 LET T1=TIME
1080 FOR A=1 TO N
1090 LET L=RND(-TIME)*5+6
1100 FOR B=1 TO L
1110 LET O$(A)=O$(A)+CHR$(INT(RND(-TIME)*26+65))
1120 NEXT B
```

```
1130 LET S$(A)=O$(A)
1140 NEXT A
1150 LET T2=TIME
1160 PRINT "Genereertijd =";(T2-T1)/50;"sec."
1170 LET T1=TIME
1179 '
1180 REM ************************
1181 REM * Eerst grootste getal *
1182 REM * zoeken. Plaats daar- *
1183 REM * van onthouden.       *
1184 REM ************************
1185 '
1190 LET G$=S$(1)
1200 LET INDEX=1
1210 FOR A=2 TO N-1
1220 IF G$<S$(A) THEN LET G$=S$(A):LET INDEX=A
1230 NEXT A
1239 '
1240 REM ************************
1241 REM * Indien grootste gro- *
1242 REM * ter dan laatste, dan *
1243 REM * omwisselen.          *
1244 REM ************************
1245 '
1250 IF G$>S$(N) THEN S$(INDEX)=S$(N): S$(N)=G$
1260 LET N=N-1
1270 IF N>=2 GOTO 1190
1280 LET T2=TIME
1290 PRINT "Sorteertijd =";(T2-T1)/50;"sec."
1299 '
1300 REM ***********************
1301 REM * Afdrukken resultaat *
1302 REM ***********************
1303 '
1310 PRINT "ongesorteerd","gesorteerd"
1320 FOR A=1 TO M
1330 PRINT O$(A),S$(A)
1340 NEXT
1350 STOP
```

Voorbeeld 6-5

```
Genereertijd = 1.52 sec.
Sorteertijd = .46 sec.
```

```
ongesorteerd  gesorteerd
OYTEOZ        IDOYJTEOJU
UFAKVFQ       LGRBMW
LGRBMW        OYTEOZ
RMXHSCN       PZKFQA
IDOYJTEOJU    RMXHSCN
PZKFQA        UFAKVFQ
VGQBWGRC      VGQBWGRC
XSCNXISD      XSCNXISD

Genereertijd = 3.68 sec.
Sorteertijd = 1.88 sec.
ongesorteerd  gesorteerd
GQBLWGRM      AKVFQLWGR
HSCNXSDNYI    DZJUEPZKF
DZJUEPZKF     GQBLWGRM
AKVFQLWGR     HRMXHSCNXS
MWRCMX        HSCNXSDNYI
SDYITDO       ITDOYJTP
TEOZKUFPKV    JUEPZUFPAK
QALVQBM       MWRCMX
HRMXHSCNXS    QALVQBM
TOZJUEP       QBLWGRBWHR
VFQALVQB      SDYITDO
WGRBXHSC      TEOZKUFPKV
XSDNYITD      TOZJUEP
JUEPZUFPAK    VFQALVQB
QBLWGRBWHR    WGRBXHSC
ITDOYJTP      XSDNYITD
```

6.5 Testbeeld

Het nu volgende programma is maar een aardigheidje. De reden om het hier op te nemen is, dat het laat zien hoe in de grafische mode toch tekst kan worden afgedrukt.

Regel 5 opent het grafische scherm voor OUTPUT onder nummer 1. Regel 10 zet de computer in de grafische mode.
De FOR...NEXT lus van regel 20, 30 en 40 zet 15 balken van allemaal verschillende kleuren op het scherm. Vervolgens zet regel 50 de pixel-cursor op positie 56,80, waarna met regel 60 de tekst B E E L D-S C H E R M over de gekleurde balken heen wordt geschreven. U ziet dat er wordt gePRINT naar nummer 1. In regel 5 was het grafisch

scherm aan nummer 1 toegekend. We printen dus naar het grafisch scherm.

Regel 70 laat een toonladder horen en regel 80 wacht tot er op de RETURN-toets wordt gedrukt, voordat er opnieuw een toonladder wordt gespeeld. U hebt nu dus de gelegenheid om zowel het scherm van de TV als de volumeregelaar daarvan af te stellen.

Listing 6-6

```
5 OPEN "grp:"FOR OUTPUT AS #1
10 SCREEN 2,1
20 FOR I=1 TO 256 STEP 256/16
30 LINE (I,0)-(I+16,192),16-I/16,BF
40 NEXT I
50 PSET (56,80)
60 PRINT #1,"T E S T B E E L D"
70 PLAY "O4CDEFGABO5C"
80 IF INKEY$<>CHR$(13) GOTO 70
90 END
```

Wie nog niet kon typen toen hij zijn MSX computer kocht, die zal heel wat tijd hebben moeten spenderen aan het leren vinden van de juiste toetsen. Het programma in dit hoofdstuk maakt dat leren vinden van de toetsen tot een plezier.

7.1 Letters zoeken

Het programma (zie listing 7–1) bedenkt een letter en geeft deze weer op het beeldscherm. De bedoeling is dat de speler dan diezelfde letter zo snel mogelijk intikt op het toetsenbord. De tijd die er tussen het op het beeldscherm afdrukken en het intikken van de letter zit, wordt gemeten. De tijd die u hebt gebruikt, wordt, evenals het beurtnummer, op het scherm afgedrukt. Dit is echter nog niet alles. Het is namelijk ook mogelijk om dit "spel" met meerdere spelers te spelen. In dat geval zal, nadat iedere speler 10 beurten heeft gehad, de winnaar worden bekend gemaakt. Daarbij worden ook alle scores van alle deelnemers afgedrukt. Dit geeft een soort spel-element aan het programma.

Wanneer u dit spel wat langer hebt gespeeld, zult u steeds beter scoren. Om het u dan weer wat moeilijker te maken zijn er drie moeilijkheidsgraden ingevoerd. De gemakkelijkste manier is, dat het in te tikken karakter steeds op een vast moment midden in het beeldscherm wordt afgedrukt. Iets moeilijker wordt het al wanneer het karakter ergens op een willekeurige plaats van de regel wordt afgedrukt. Het moeilijkste is echter om te reageren op het verschijnen van een karakter dat op een willekeurig moment op een willekeurige plaats van het beeldscherm verschijnt.

Listing 7-1

```
10 LET VARGEN=100
20 GOTO 1000
100 LET X=INT(RND(1)*20)
110 LET Y=INT(RND(1)*40)
140 LET W=INT(RND(-TIME)*2000)
150 LET L=INT(RND(-TIME)*26)+65
200 IF M=3 THEN RETURN
210 LET W=100:LET X=10
220 IF M=2 THEN RETURN
230 LET Y=15
```

```
240 RETURN
1000 CLS:PRINT "Het spel LETTERZOEKEN kan met meer
dere  spelers worden gespeeld."
1010 PRINT "Geef het aantal spelers dat mee doet o
p"
1020 INPUT "Aantal spelers";S
1030 PRINT "Het aantal spelers is";S
1035 IF S<1 THEN PRINT "Bedankt voor het spelen en
 tot ziens":END
1040 PRINT "Aantal goed (j/n)";
1050 INPUT A$
1060 IF A$<>"j" AND A$<>"J" GOTO 1000
1070 CLS
1100 DIM N$(S):DIM T(S)
1110 PRINT "Ik zal u nu gaan vragen hoe alle spele
rsheten."
1120 FOR A=1 TO S
1130 PRINT "Hoe heet speler ";A;
1140 INPUT N$(A)
1150 PRINT N$(A);" is speler";A
1160 NEXT
1170 FOR A=1 TO 500:NEXT
1200 CLS:PRINT "Dit spelletje kan op 3 manieren wo
rden  gespeeld."
1210 PRINT "1 = het gemakkelijkste"
1220 PRINT "2 = de letter verschijnt steeds  op ee
n andere plaats op de zelfde regel"
1230 PRINT "3 = de letter vershcijnt steeds op een
  andere plaats op steeds andere   regels."
1240 PRINT "Kies een van de hierboven genoemde
  spelsoorten."
1250 INPUT "Spelsoort (1, 2 of 3)";M
1260 PRINT "Je hebt gekozen voor spelsoort nummer"
;M
1270 FOR A=1 TO 500:NEXT
1280 CLS
1300 FOR Z=1 TO S
1310 PRINT "De beurt is aan ";N$(Z)
1320 PRINT "Als je op de ENTER-toets drukt, zal ik
  het scherm schoonmaken. Daarna zal ik   een lett
er laten verschijnen. Je moet   proberen die lette
r zo vlug  mogelijk   in te drukken."
1330 PRINT "Ik zal je dan vertellen hoe vlug je  h
ethebt gedaan, door je een aantal  punten te geven
."
```

```
1340 PRINT "Hoe minder punten, hoe beter. Je krijg
t 10 beurten. Daarna is de volgende speleraan de b
eurt."
1350 PRINT "Doe je best "; N$(Z)
1360 INPUT "Druk op de ENTER-toets om te beginnen.
";T$
1370 CLS
1380 FOR C=1 TO 10
1410 GOSUB 100
1420 FOR A=0 TO W:NEXT
1425 LOCATE Y,X
1430 PRINT CHR$(L)
1440 T1=TIME
1450 A$=INKEY$
1460 IF A$="" GOTO 1450
1480 IF ASC(A$)-32=L THEN BEEP
1490 IF ASC(A$)-32<>L THEN 1450
1500 T2=TIME
1505 IF T1>T2 THEN T1=T2
1510 LET T(Z)=T(Z)+(T2-T1)
1520 PLAY "t255ceg"
1525 IF PLAY(0)<>0 GOTO 1525
1530 LOCATE Y,X: PRINT " "
1535 LOCATE 0,20: PRINT "beurt ";C
1540 PRINT "tijd =";T2-T1,"totaal =";T(Z)
1550 NEXT C
1555 PLAY "t255gegc"
1556 IF PLAY(0)<>0 GOTO 1556
1560 CLS
1570 NEXT Z
1600 LET LAAG=T(1):LET L$=N$(1)
1605 IF S=1 GOTO 1650
1610 FOR A=2 TO S
1620 IF T(A)<LAAG THEN LET L$=N$(A)
1630 IF T(A)<LAAG THEN LET LAAG=T(A)
1640 NEXT A
1650 CLS
1660 PRINT L$;" heeft gewonnen"'l$;" had ";laag;"
punten"''
1700 FOR A=1 TO S
1710 PRINT N$(A);" heeft ";T(A);' punten
1720 NEXT A
1800 FOR A=100 TO 1000 STEP 5
1810 P=INT(1789770#/(16*A))
```

```
1820 R1=INT(P/256)
1830 R0=P-R1*256
1840 SOUND 0,R0:SOUND 1,R1
1850 SOUND 7,62:SOUND 8,15
1860 NEXT A
1870 SOUND 7,63
1900 RUN
```

Het aantal spelers kunt u kiezen in regel 1020. Vervolgens wordt iedere speler gevraagd zijn naam in te geven (regels 1120 tot en met 1160). Van regel 1200 tot en met 1260 wordt de moeilijkheidsgraad gevraagd. De FOR NEXT lus op regel 1300 wordt voor iedere speler 1 keer uitgevoerd. Regels 1310 tot en met 1360 geven wat informatie betreffende het spel. De FOR NEXT lus op regel 1400 wordt voor iedere speler 10 keer uitgevoerd.

Met regel 1410 wordt de subroutine op regel 100 uitgevoerd. In die subroutine worden willekeurige waarden in variabelen gezet. Deze subroutine start op regel 100. De gegenereerde variabelen zullen later in het spel worden gebruikt. De FOR NEXT lus van regel 1400 gaat door tot regel 1550. Ik zal de regels niet stuk voor stuk bespreken.

Regel 1570 is het einde van de spelbeurt van een speler. Hebben alle spelers een beurt gehad, dan zal worden verder gegaan met regel 1600. Dit is een u bekende routine, het zoeken naar het kleinste getal uit een rij. Vanaf regel 1660 worden de resultaten van het spel afgedrukt. Om de spelers in de gelegenheid te stellen die resultaten te lezen wordt er met regels 1800 tot en met 1870 een sirene ten gehore gebracht.

8. LETTERSPELLETJES VOOR DE KLEINTJES

Sinds ik mijn computer, nu ongeveer een jaar geleden, kreeg, is het mij opgevallen dat mijn jongste dochter niets liever deed dan letters intikken. Wat letters precies waren wist ze niet, maar ze had heel goed door dat het erg belangrijk was als je letters kon maken. En dit was op de computer erg gemakkelijk.

Dit bracht mij op het idee, speciaal voor haar en voor alle andere kinderen van haar leeftijd (ze is net 5 jaar geworden), een spelletje met letters te maken. Omdat ze nog niet kan lezen moet het nog tamelijk eenvoudig blijven. Wat ik in het volgende spelletje doe is, een groot aantal gelijke letters op het scherm afdrukken, waarna er temidden van al die letters 1 afwijkende letter tussen wordt gezet. Nu moet het kind de cursor (het witte vakje) met behulp van de pijltjestoetsen naar die afwijkende letter verplaatsen. Staat de cursor op die afwijkende letter, dan moet het kind op de RETURN-toets drukken.

Het is mij gebleken dat kinderen van de kleuterschoolleeftijd het programma binnen een minuut kunnen bedienen. Nog veel belangrijker echter is, dat die kinderen het een prachtig spelletje vinden. Jammer genoeg ben ik geen pedagoog, en ik weet dus niet of het wel echt leerzaam voor ze is.

8.1 Letters voor wie nog niet kan lezen

Het hiervoor besproken spel is in listing 8–1 weergegeven. Hier volgt een korte beschrijving. Het programma begint op regel 90. Regel 100 zet een willekeurige letter in variabele Z$. Daarna wordt met regel 110 de eerstvolgende letter uit het alfabet, of de letter A indien in Z$ de Z stond, in X$ gezet. Nu wordt er met behulp van regel 190 tot en met 340 een vierkant gemaakt. Met regels 400 tot en met 440 worden 25 gelijke en één afwijkende letter in het vierkant geschreven.
Met regel 520 wordt de cursor midden in het vierkant geschreven. Regel 840 leest welk karakter op de cursor positie staat, en slaat dat karakter op in L$.

Listing 8-1

```
10    '*********************************
20    '* ZOEK DE GEVRAAGDE LETTER OP *
30    '* TUSSEN DE IN DE RECHTHOEK    *
```

```
40  '* AFGEBEELDE LETTERS, DOOR ER *
50  '* DE CURSOR OP TE ZETTEN EN    *
60  '* DRUK DAARNA OP RETURN.       *
70  '*******************************
80  '
90 WIDTH 40:CLS
100 Z$=CHR$(RND(-TIME)*26+65)
110 IF Z$=>"Z" THEN X$="A" ELSE X$=CHR$(ASC(Z$)+1)
120 PRINT "Zet de cursor op de letter die anders is"
130 PRINT "druk daarna op RETURN."
140 '
150 '*******************************
160 '*  PRINT VIERKANT OP SCHERM    *
170 '*******************************
180 '
190 Y=4
200 FOR X=9 TO 31
210 LOCATE X,Y:PRINT "*";
220 NEXT X
230 X=31
240 FOR Y=5 TO 20
250 LOCATE X,Y:PRINT "*";
260 NEXT Y
270 Y=21
280 FOR X=31 TO 9 STEP -1
290 LOCATE X,Y:PRINT "*";
300 NEXT X
310 X=9
320 FOR Y=20 TO 5 STEP -1
330 LOCATE X,Y:PRINT "*";
340 NEXT Y
350 '
360 '**********************************
370 '*   ZET LETTERS OP HET SCHERM     *
380 '**********************************
390 '
400 FOR A=1 TO 25
410 LOCATE RND(1)*21+10,RND(1)*16+5
420 PRINT Z$;
430 NEXT A
440 LOCATE RND(1)*21+10,RND(1)*16+5: PRINT X$
450 '
460 '**********************************
470 '* SCHRIJF EN VERPLAATS DE CURSOR *
```

```
480 '* AFHANKELIJK VAN DE INGAVE.        *
490 '**********************************
500 '
510 X=20:Y=13
520 LOCATE X,Y,1
530 I$=INKEY$
540 IF I$=CHR$(30) THEN GOSUB 660:'up
550 IF I$=CHR$(28) THEN GOSUB 690:'right
560 IF I$=CHR$(31) THEN GOSUB 720:'down
570 IF I$=CHR$(29) THEN GOSUB 750:'left
580 IF I$=CHR$(13) GOTO 840: RETURN
590 IF I$=CHR$(27) GOTO 90: ESC
600 GOTO 520
610 '
620 '**********************************
630 '* SUBROUTINES CURSORVERPLAATSING *
640 '**********************************
650 '
660 Y=Y-1
670 IF Y<5 THEN Y=20
680 RETURN
690 X=X+1
700 IF X>30 THEN X=10
710 RETURN
720 Y=Y+1
730 IF Y>20 THEN Y=5
740 RETURN
750 X=X-1
760 IF X<10 THEN X=30
770 RETURN
780 '
790 '**********************************
800 '* LEES EN VERGELIJK DE ONDER DE  *
810 '* CURSOR STAANDE LETTER.         *
820 '**********************************
830 '
840 LOCATE 0,23: L$=CHR$(VPEEK(BASE(0)+X+Y*40))
850 IF L$<>X$ THEN PLAY "T255O6CO5CO4CO3C":GOTO 520
860 RESTORE 910
870 READ P$
880 PLAY P$
890 IF PLAY(0)<>0 GOTO 890
900 GOTO 10
910 DATA "O4CDEFGABO5C"
920 END
```

Voorbeeld 8-1

```
Zet de cursor op de letter die anders
is druk daarna op RETURN.

        ***********************
        *                     *
        *O          O      O  *
        *   O                 *
        *     P   O   O O   O *
        *                     *
        *     O       O       *
        *              O      *
        *          O        O *
        *O                   O*
        *     O   O           *
        *                     *
        *     O O             *
        *                     *
        *   O O              O*
        *   O         O       *
        *                     *
        ***********************
```

Nu kan het spel beginnen. Het kind zoekt de afwijkende letter binnen het vierkant op en bepaalt welke kant het zwarte blokje op moet. De regels 530 tot en met 590 onderzoeken het ingetypte karakter. De verschillende routinetjes voor links, op, neer en rechts dienen ervoor om te voorkomen dat de cursor uit het vierkant loopt.

Denkt het kind de cursor op de afwijkende letter te hebben gezet, dan zal het op RETURN drukken. Daarmee wordt naar regel 840 gesprongen. Deze regel onderzoekt welk karakter er op de plaats van de cursor staat. Is dat inderdaad het afwijkende karakter, dan wordt met regel 880 een liedje gemaakt. U ziet daar de RESTORE-instructie staan. Met RESTORE wordt een wijzer op het eerste data-element uit alle data-elementen gezet.
Het gevolg daarvan is dat de wijzer op het eerste data-element uit de DATA-regel komt te staan.

Regel 870 bevat de READ-instructie. Hiermee wordt dus het eerste data-element uit een regel gelezen.

Dit DATA-element is het met PLAY te spelen liedje.

De belangrijkste techniek die in dit programma is gebruikt, en die ook al in de voorgaande programma's werd gebruikt, is het gebruik van VPEEK. U ziet dat het mogelijk is om iedere gewenste positie van het beeldscherm daarmee uit te lezen. Afhankelijk van het gelezen karakter kan dan een bepaalde actie worden ondernomen. Dit geeft veel interessante mogelijkheden.

8.2 Zoek een opgegeven letter uit het alfabet

In dit programma wordt de in Z$ te plaatsen letter gebruikt om door het kind te laten zoeken. Regels 410 tot en met 440 zetten nu 26 verschillende letters in het vierkant.

Er schuilt nu echter een programma-addertje onder het gras. Door de letters van het alfabet op een willekeurige plaats te schrijven, is het mogelijk dat op een gegeven moment een letter over een al eerder geschreven letter heen wordt afgedrukt. In dat geval zal de eerst afgedrukte letter niet meer op het scherm voorkomen. Stel nu dat het kind de opdracht krijgt om juist die letter, die er niet meer is, op te zoeken; het zou dan nooit het goede antwoord kunnen geven. Om deze situatie te voorkomen, kan het kind zeggen dat de gevraagde letter niet op het scherm voorkomt door de ESCAPE-toets in te drukken. Hiervoor is regel 590 er opgenomen.

Voor het overige is het programma praktisch gelijk aan het vorige. Hopelijk zult u nu inzien dat met enkele kleine wijzigingen allerlei moeilijkheidsgraden kunnen worden gemaakt. Op die manier zal het programma nog lange tijd interessant blijven voor het kind. Ook het uitbreiden van de liedjes verdient aanbeveling. Probeer zo mogelijk die liedjes te maken die het kind zojuist op school heeft geleerd.

Op regels 990 tot en met 1040 ziet u de routine die een copie van het beeldscherm kan maken op een printer. Door van regel 920 GOTO 990 te maken wordt steeds een afdruk van het beeldscherm verkregen. Zo zijn ook de voorbeelden 8-1 en 8-2 gemaakt.

Listing 8-2

```
10   '**********************************
20   '* ZOEK DE GEVRAAGDE LETTER OP *
30   '* TUSSEN DE IN DE RECHTHOEK    *
```

```
40  '* AFGEBEELDE LETTERS, DOOR ER *
50  '* DE CURSOR OP TE ZETTEN EN    *
60  '* DRUK DAARNA OP RETURN.       *
70  '*******************************
80  '
90 KEY OFF
100 WIDTH 40:CLS
110 Z$=CHR$(RND(-TIME)*26+65)
120 K=INT(RND(-TIME)*4+1):COLOR K,K+1
130 PRINT "Zet de cursor op de letter ";Z$
140 PRINT "druk daarna op RETURN."
150 '
160 '*******************************
170 '*  PRINT VIERKANT OP SCHERM   *
180 '*******************************
190 '
200 Y=4
210 FOR X=9 TO 31
220 LOCATE X,Y:PRINT "*";
230 NEXT X
240 X=31
250 FOR Y=5 TO 20
260 LOCATE X,Y:PRINT "*";
270 NEXT Y
280 Y=21
290 FOR X=31 TO 9 STEP -1
300 LOCATE X,Y:PRINT "*";
310 NEXT X
320 X=9
330 FOR Y=20 TO 5 STEP -1
340 LOCATE X,Y:PRINT "*";
350 NEXT Y
360 '
370 '**********************************
380 '*    ZET LETTERS OP HET SCHERM    *
390 '**********************************
400 '
410 FOR L=65 TO 90
420 LOCATE RND(1)*21+10,RND(1)*16+5
430 PRINT CHR$(L);
440 NEXT L
450 '
460 '**********************************
470 '* SCHRIJF EN VERPLAATS DE CURSOR *
```

```
480 '* AFHANKELIJK VAN DE INGAVE.        *
490 '**********************************
500 '
510 X=20:Y=13
520 LOCATE X,Y,1
530 I$=INKEY$
540 IF I$=CHR$(30) THEN GOSUB 660:'up
550 IF I$=CHR$(28) THEN GOSUB 690:'right
560 IF I$=CHR$(31) THEN GOSUB 720:'down
570 IF I$=CHR$(29) THEN GOSUB 750:'left
580 IF I$=CHR$(13) GOTO 840:'gevonden
590 IF I$=CHR$(27) GOTO 100:'weet niet
600 GOTO 520
610 '
620 '**********************************
630 '* SUBROUTINES CURSORVERPLAATSING *
640 '**********************************
650 '
660 Y=Y-1
670 IF Y<5 THEN Y=20
680 RETURN
690 X=X+1
700 IF X>30 THEN X=10
710 RETURN
720 Y=Y+1
730 IF Y>20 THEN Y=5
740 RETURN
750 X=X-1
760 IF X<10 THEN X=30
770 RETURN
780 '
790 '**********************************
800 '* LEES EN VERGELIJK DE ONDER DE   *
810 '* STAANDE LETTER.                  *
820 '**********************************
830 '
840 LOCATE 0,23: L$=CHR$(VPEEK(BASE(0)+X+Y*40))
850 IF L$<>Z$ THEN PLAY "T255l8O6CO5CO4CO3C":GOTO 520
860 FOR I=0 TO 95 STEP 5
870 I$=STR$(I):L=LEN(I$):I$=RIGHT$(I$,L-1)
880 PLAY "t255l64n"+I$
890 COLOR ,I/10+1
900 IF PLAY(0)<>0 GOTO 900
910 NEXT I
```

```
920 GOTO 10
930 '
940 '**********************************
950 '* routine voor het afdrukken van *
960 '* beeldscherminhoud naar printer *
970 '**********************************
980 '
990 FOR J=0 TO 21
1000 FOR I=0 TO 39
1010 LPRINT CHR$(VPEEK(BASE(0)+J*40+I));
1020 NEXT I
1030 LPRINT
1040 NEXT J
1050 END
```

Voorbeeld 8-2

```
Zet de cursor op de letter O
druk daarna op RETURN.

* * * * * * * * * * * * * * * * * * * * * * *
*               Z         U                 *
*                   N     Y                 *
*             H                             *
*       J                                   *
*                     G                     *
*           V               F               *
* M                                         *
*                     B                     *
*           I                               *
*                       A         T         *
*                                           *
*   D S   C                                 *
*                                       W   *
*                             P             *
*       X           K                     Q *
*       O                                 R *
* * * * * * * * * * * * * * * * * * * * * * *
```

9. ANAGRAM

Wanneer je de letters van een woord willekeurig door elkaar gooit, blijkt hoeveel verschillende combinaties er met die letters mogelijk zijn. Zelfs met de letters van eenvoudige woorden zijn al ontelbaar vele combinaties te maken. Het gevolg hiervan is, dat het moeilijk is om, wanneer je die door elkaar gezette letters voor je neus krijgt, het orginele woord terug te vinden. Dit gegeven nodigt uit tot het maken van een computerspelletje.

Laat de computer een willekeurig woord nemen, daarvan de letters door elkaar zetten, en afdrukken op het beeldscherm. U mag nu raden welk woord wordt bedoeld. Toen ik dit spelletje voor het eerst ging spelen, bleek al gauw dat het soms veel te moeilijk was. Ik had zo nu en dan hulp nodig. Het was vaak al een hele hulp als je de eerste letter wist. Maar zelfs dat was soms nog niet genoeg. Het spel is daarom nu zo gemaakt dat je de computer op ieder moment om hulp kunt vragen. Iedere keer echter dat je hulp krijgt, wordt er een puntje van je score afgetrokken.

U ziet het, het principe van het spel is eenvoudig. Laten we maar eens kijken of het programma ook zo eenvoudig is. Het programma is weergegeven in listing 9–1. Het eigenlijke begin van het programma is regel 1000. Regels 1000 tot en met 1040 drukken informatie over de bedoeling van het spel op het beeldscherm af. Regels 1050 tot en met 1110 wachten op het intoetsen van de RETURN-toets. Zolang echter die toets nog niet wordt ingedrukt, wordt geluid geproduceerd. Dit blijft doorgaan totdat de RETURN-toets is ingedrukt.

Op dat moment kan het spel pas echt beginnen. Het scherm wordt schoongemaakt en de puntentelling wordt op nul gezet. Hierna begint een FOR...NEXT lus die tien keer zal worden doorlopen. In iedere lus krijgt u een woord te raden. De regels 1171, 1172 en 1173 bepalen welk woord u moet gaan raden. De DATA-regels starten op regelnummer 5000. Met de RESTORE-instructie zetten we de pointer op het eerste item van de eerste DATA-regel. Vervolgens genereren we een willekeurig getal. En met regel 1173 lezen we net zo vaak een item uit een DATA-regel als het willekeurige getal aangeeft. Na iedere leesactie wordt de pointer op een volgende DATA-regel gezet. We doen niets met de gelezen informatie. Het enige wat we bereiken is, dat we de pointer op een willekeurige DATA-regel zetten. Hierna gaan we verder met het programma.

Met regel 1180 lezen we het te raden woord en het bijbehorende onderwerp. Stel nu dat het willekeurige getal er voor heeft gezorgd dat we DATA-regel 5023 met de leesopdracht in regel 1180 hebben gelezen, dan zal nu de variabele w$ het woord „volvo" bevatten, en de variabele h$ het woord „automerk". De letters van het woord „volvo" moeten nu door elkaar gegooid worden. De dan ontstane reeks van letters wordt opgeslagen in r$. De routine van regel 1190 tot en met 1380 doet dit voor ons. Dat deze routine zo ingewikkeld is, komt doordat we moeten voorkomen dat de letters in r$ over elkaar heen worden gezet.

Vanaf regel 1400 kunnen we beginnen het te raden woord af te drukken. Regel 1410 zorgt er voor dat dat woord netjes gecentreerd op regel 3 van het beeldscherm terecht komt. Vanaf regel 1430 kunt u dan de letters van het te raden woord intikken. Regel 1435 zorgt ervoor dat de score wordt afgedrukt. De ingetikte letter wordt in l$ bewaard. Regel 1470 en 1480 kijken of er in l$ een spatie staat. Is dat het geval dan hebt u blijkbaar om hulp gevraagd. Nu is het belangrijk of u al eerder om hulp had gevraagd, want de eerste keer dat u om hulp vraagt zal de categorie waarin het te raden woord valt worden afgedrukt (de categorie staat in variabele h$). Vraagt u daarna nog eens om hulp, dan zal de eerstvolgende letter van het te raden woord worden afgedrukt. Regel 1490 zorgt er voor dat de computer weet of het de eerste of een volgende vraag om hulp was. Regel 1500 zorgt er voor dât u na het vragen van hulp weer een nieuwe letter kunt indrukken.

Regel 1530 kijkt of de ingegeven letter de juiste letter is.
Is dit inderdaad zo, dan wordt een vreugdekreetje geslaakt, een punt bij de score geteld, en de score op het beeldscherm afgedrukt. Werd echter de verkeerde letter ingedrukt, dan wordt dat met regel 1540 aan de speler gemeld. Bovendien wordt met regels 1560 tot en met 1570 een treurig geluid gemaakt. Daarna haalt regel 1575 de foutboodschap weer van het scherm en wordt het intikken van de volgende letter afgewacht (volgende doorgang van de FOR NEXT lus van regel 1430).

Zodra alle letters zijn ingetikt, wordt met regel 1600 een vreugdekreet geslaakt, waarna de volgende beurt kan worden gestart (de volgende doorgang van de FOR NEXT lus van regel 1170). Zelfs al zou nu het zelfde woord uit de DATA-regels worden gelezen, dan nog is de kans klein dat dit er hetzelfde uitziet als de vorige keer. Immers, de letters worden willekeurig door elkaar gezet. Dit maakt het mogelijk om met

een relatief klein aantal woorden een zeer groot aantal verschillende raadsels te maken.

Na 10 beurten is het spel afgelopen. Regel 1640 geeft u de mogelijkheid om het spel nog eens te spelen. Het staat u natuurlijk vrij om zelf nieuwe woorden te bedenken en in DATA-regels te zetten. Denk er daarbij om dat het eerste woord in de DATA-regel steeds het te raden woord is, terwijl het tweede woord de categorie is waaronder het eerste woord valt. Maak de te raden woorden niet te lang, ze worden dan te moeilijk om te raden. Zelfs woorden van vijf letters zijn soms al moeilijk genoeg.

Listing 9-1

```
10 DIM I(20)
100 GOTO 1000
200 LOCATE 0,21:PRINT "                                    "
;
210 LOCATE 0,21:PRINT "Je hebt nu";PUNTEN;"punten"
;
220 RETURN
500 PLAY "t255l64cceegger16egfedcccc"
510 IF PLAY(0)<>0 GOTO 510
530 RETURN
1000 CLS:PRINT "           A N A G R A M"
1010 PRINT "In het nu volgende spelletje zullen  w
oorden op het scherm verschijnen,   waarvan alle l
etters door elkaar     staan."
1020 PRINT "Als je een woord niet weet, druk dan o
p de SPATIE-BALK."
1030 PRINT "Je krijgt dan wat hulp, al wordt er  w
el een punt afgetrokken."
1040 PRINT "Druk op RETURN om te beginnen"
1050 LET B=0
1060 LET B=B+1
1070 IF B=7 THEN LET B=0
1080 PLAY CHR$(B+65)
1090 IF PLAY(0)<>0 GOTO 1090
1100 LET A$=INKEY$
1110 IF A$<>CHR$(13) GOTO 1060
1130 CLS
1140 LET PUNTEN=0
1170 FOR Z=1 TO 10
```

```
1171 RESTORE
1172 R=INT(RND(-TIME)*50)+1
1173 FOR I=1 TO R:READ W$,H$:NEXT I
1175 LOCATE 0,20:PRINT "Beurtnummer";Z;
1180 READ W$,H$
1190 LET L=LEN(W$)
1200 A$=""
1220 LET R$=""
1250 FOR A=1 TO L
1260 LET I(A)=INT(RND(1)*L)+1
1270 IF A=1 THEN 1350
1280 LET B=0
1290 FOR C=1 TO A-1
1300 IF I(A)<>I(C) THEN 1330
1310 LET B=1
1320 LET C=A
1330 NEXT C
1340 IF B=1 THEN 1260
1350 NEXT A
1360 FOR A=1 TO L
1370 LET R$=R$+MID$(W$,(I(A)),1)
1380 NEXT A
1390 LET H=0
1400 LOCATE 0,0:PRINT "Het te raden woord is:"
1405 LOCATE 10,3:PRINT "                    "
1410 LOCATE 17-L/2,3:PRINT R$
1415 LOCATE 0,6
1420 PRINT "De ingetikte letters zijn:"
1425 LOCATE 19-(L+4)/2,9:PRINT "                    "
1430 FOR A=1 TO L
1435 GOSUB 200
1440 LET L$=INKEY$
1450 IF L$="" GOTO 1440
1465 LOCATE 0,12:PRINT "
          "
1470 IF L$=" " AND H=1 THEN LOCATE 18-(L+4)/2+A,9:
PRINT MID$(W$,A,1);:A$=A$+MID$(W$,A,1):LET PUNTEN=
PUNTEN-1:GOTO 1590
1480 IF H=O AND L$=" " THEN LOCATE 0,12:PRINT "OND
ERWERP: ";H$:LET PUNTEN=PUNTEN-1:LET H=1
1485 GOSUB 200
1490 IF L$=" " AND H=0 THEN LET H=1
1500 IF L$=" " THEN 1440
1510 IF L$=MID$(W$,A,1) THEN A$=A$+L$
```

```
1520 LOCATE 19-(L+4)/2,9:PRINT A$
1530 IF L$=MID$(W$,A,1) THEN BEEP: PUNTEN=PUNTEN+1
:GOSUB 200:GOTO 1590
1540 LOCATE 15,15:PRINT "FOUT"
1560 PLAY "t64l16bagfedc"
1570 IF PLAY(0)<>0 GOTO 1570
1575 LOCATE 15,15:PRINT "       "
1580 LET A=L
1590 NEXT A
1600 GOSUB 500
1610 LOCATE 16-L/2,9:PRINT "                 "
1620 NEXT Z
1630 CLS
1640 PRINT "Als je nog een spelletje wilt, type  d
an de letter j gevolgd door de      ENTER-toets."
1650 INPUT X$
1660 IF X$<>"j"AND X$<>"J" THEN END
1670 RUN
5000 DATA "tafel","meubelstuk"
5001 DATA "marije","naam"
5002 DATA "monique","naam"
5003 DATA "bier","drank"
5004 DATA "sherry","drank"
5005 DATA "ketel","in de keuken"
5006 DATA "whisky","drank"
5007 DATA "limonade","drank"
5008 DATA "marion","naam"
5009 DATA "ernst","naam"
5010 DATA "stoel","meubelstuk"
5011 DATA "hond","beest"
5012 DATA "paard","beest"
5013 DATA "schaap","beest"
5014 DATA "rakker","hondenaam"
5015 DATA "potlood","schrijfgerei"
5016 DATA "varken","beest"
5017 DATA "televisie","huiskamer"
5018 DATA "inge","naam"
5019 DATA "michael","naam"
5020 DATA "bank","meubelstuk"
5021 DATA "leeuw","beest"
5022 DATA "tijger","beest"
5023 DATA "volvo","automerk"
5024 DATA "lancia","automerk"
5025 DATA "gitaar","muziekinstrument"
```

```
5026 DATA "geiser","in de keuken"
5027 DATA "fornuis","in de keuken"
5028 DATA "soep","voedsel"
5029 DATA "belgie","land"
5030 DATA "kruk","meubelstuk"
5031 DATA "zweden","land"
5032 DATA "spanje","land"
5033 DATA "italie","land"
5034 DATA "denemarken","land"
5035 DATA "vulpen","schrijfgerei"
5036 DATA "computer","?"
5037 DATA "eikeboom","plant"
5038 DATA "berkeboom","plant"
5039 DATA "grasveld","tuin"
5040 DATA "kast","meubelstuk"
5041 DATA "thee","drank"
5042 DATA "koffie","drank"
5043 DATA "wessel","naam"
5044 DATA "volkswagen","automerk"
5045 DATA "kroket","voedsel"
5046 DATA "zwitserland","land"
5047 DATA "duitsland","land"
5048 DATA "noorwegen","land"
5049 DATA "engeland","land"
5050 DATA "divan","meubelstuk"
```

10. SPRITES DEFINIEREN EN OPSLAAN

Het is heel goed mogelijk om SPRITES met de hand, op een stukje papier te maken. Je kunt hiervoor gebruik maken van een ruitjesblok. Het is echter nogal een klus om de zo gedefiniëerde sprites om te zetten in een aantal getallen. Dit wordt vooral moeilijk bij sprites van 16x16 pixels. De kans op vergissingen is bovendien ook groot, met het gevolg dat de sprite er niet precies zo uit ziet als je graag zou willen. Dit was voor mij aanleiding om een programma te maken waarmee ik op het beeldscherm sprites kon definiëren.

Het op die manier definiëren van sprites is een ding, het vervolgens kunnen gebruiken van die sprites is een ander ding. Om de gedefiniëerde sprites in een ander programma te kunnen gebruiken, leek het mij een goed idee om de gedefiniëerde sprites op te bergen in een file op cassette. Het programma dat de sprites dan wil gebruiken zal die cassette file dan moeten lezen, en de gewenste sprites met de daartoe beschikbare statements in het video RAM opbergen.

In dit hoofdstuk zullen dan ook twee programma's worden gegeven; eerst een programma om sprites mee te definiëren en op cassette op te slaan, dan een programma waarmee de sprites van cassette worden gelezen en op het beeldscherm weergegeven. De functie van dit laatste programma is alleen om u te laten zien hoe de cassette file kan worden uitgelezen en hoe de gelezen sprites op het beeldscherm kunnen worden afgedrukt.

10.1 Definiëren van sprites

Het programma uit listing 10.1 is in feite erg kort. Het loopt door van regel 100 tot regel 390. Eenmaal op regel 390 aangekomen, wordt gewacht tot er een functietoets is ingedrukt. Het indrukken van een functietoets veroorzaakt de uitvoering van een subroutine. Er zijn subroutines voor het „aan" zetten van een pixel, het „uit" zetten van een pixel, het naar links en naar boven gaan van de cursor en het opslaan van een gedefiniëerde sprite op cassette. Daarnaast is er nog een subroutine die een sprite-array definiëert en die op het scherm een vergrote weergave van de sprite geeft. Hoe een en ander werkt zullen we hierna zien.

Na het starten van het programma wordt u gevraagd een lege cassette in de recorder te zetten, en de recorder op opname te zetten. Nu

wordt met regel 150 een cassette file geopend.

Daar we de functietoetsen zullen gebruiken om de sprite te definiëren, moeten nu eerst de nieuwe functies aan die toetsen worden toegekend. Dit gebeurt in regels 220 tot en met 260, terwijl regel 270 er voor zorgt dat de nieuwe functies op de 24ste regel van het scherm worden afgedrukt. Regel 280 activeert de functietoetsen, en regel 290 zorgt er voor dat de computer weet welke subroutine moet worden uitgevoerd na het indrukken van een functietoets.

Regel 370 initialiseert een array van 16 x 16 numerieke velden. In deze array (r$) zullen de waarden van de gedefiniëerde sprite worden opgeborgen. Daarna wordt naar de subroutine op regel 1600 gegaan. Regels 1600 tot en met 1640 vragen u de grootte van de straks te definiëren sprite. U mag 8x8 en 16x16 sprites door elkaar maken. De grootte van de sprite zal namelijk worden meegeschreven naar de cassette file, zodat later, wanneer de file wordt gelezen, gemakkelijk kan worden nagegaan hoe groot de gelezen sprite zal zijn. De array s$ wordt vervolgens met regel 1700 tot en met 1740 op 0 gezet. Daarna wordt met de regels 1800 tot en met 1870 een vergrote sprite op het beeldscherm afgedrukt. Om aan te geven dat er op dit moment nog geen enkele pixel van de sprite is aangezet, wordt ieder pixel door het „–" teken weergegeven.

Bij terugkomst in het hoofdprogramma, op regel 390, blijft het programma in een lus zitten, waarbij steeds van regel 390 naar zichzelf wordt gesprongen. De enige manier om deze lus te onderbreken is het indrukken van een functietoets.

Drukt u functietoets 1 in, dan zal de routine op regel 450 worden uitgevoerd. Daarin wordt het pixel (op het beeldscherm) waar de cursor op staat, gezet. U ziet dan dat het „–" teken een blokje wordt. Bovendien wordt de overeenkomstige positie binnen de array s$ op 1 gezet. Tenslotte wordt de cursor naar het volgende pixel verplaatst en wordt teruggekeerd naar regel 390.

Drukt u op functietoets 2, dan zal de routine op regel 570 worden gestart. In die routine wordt het pixel waar de cursor op staat, ge-reset. U ziet dan dat, ook als daar wat anders stond, op de plaats van de cursor een „–" teken verschijnt. De cursor wordt op het volgende pixel gezet en de overeenkomstige plaats in array s$ wordt op 0 gezet, waarna wordt teruggekeerd naar regel 390.

Hebt u zich nu ergens vergist, dan kunt u door het indrukken van functietoets 3 de cursor naar links verplaatsen. Door het indrukken van functietoets 4 wordt de cursor naar boven verplaatst. Hierna kunt u de pixels weer zetten of terugzetten met de functietoetsen 1 en 2.

Als de sprite er precies zo uit ziet zoals u dat graag wilt, dan drukt u op functietoets 5. Hiermee wordt de subroutine op regel 850 gestart. In deze subroutine wordt eerst gekeken of het om een 8x8 dan wel een 16x16 sprite gaat. Afhankelijk hiervan zal naar regel 960 of regel 1110 worden gesprongen. Na uitvoering van de 8x8 of 16x16 routines zullen de regels 1490 en volgenden weer voor beide soorten sprites gezamenlijk gelden.

In de 8x8 routine wordt de array s$ uitgelezen en worden de bitjes samengesteld tot een waarde, die kan variëren van 0 tot 255. Dit moet 8 keer gebeuren; er zijn immers 8 rijen van 8 bitjes.

In de 16x16 routine gebeurt in principe precies hetzelfde, alleen zijn er nu 4 keer zoveel bitrijen te bewerken. De volgorde van bewerking ligt vast en blijkt uit de volgende afbeelding:

16x16 = 4 x 8x8

1	3
2	4

Als u met regel 1500 beslist dat u nog meer sprites wilt definiëren, dan wordt naar regel 1600 gesprongen. Daar begint het hele proces opnieuw. Beslist u echter dat u wilt stoppen, dan wordt doorgegaan met regel 1530. Daarin wordt u gevraagd even geduld te hebben. De gegenereerde sprite file moet namelijk nog naar cassette worden geschreven. Normaal gesproken doe je dit met een CLOSE statement, doch het mag ook met een END statement. Het END statement sluit namerlijk ook alle files af. Pas nadat de file helemaal op cassette staat zal de computer de systeemboodschap „Ok" geven. U hebt nu de door u gedefiniëerde sprites in een file op cassette staan.

Listing 10-1

```
10 '***********************************
20 '* MET DIT PROGRAMMA KUNNEN SPRITES*
30 '* WORDEN GEDEFINIEERD EN NAAR EEN *
```

```
40 '* CASSETTE-FILE WORDEN GESCHREVEN.*
50 '**********************************
60 '*  CASSETTERECORDER KLAARZETTEN   *
70 '*  EN SPRITE-FILE OPENEN.         *
80 '**********************************
90 '
100 WIDTH 40:CLS
110 PRINT "PLAATS EEN LEGE CASSETTE IN DE RECORDER
,ZET DE RECORDER IN DE OPNAMESTAND EN    DRUK DAAR
NA OP DE RETURN-TOETS"
120 IF INKEY$<>CHR$(13) GOTO 120
130 PRINT
140 PRINT "SPRITE BESTAND, SPRFIL, WORDT GEOPEND"
150 OPEN "CAS:SPRFIL" FOR OUTPUT AS #1
160 '
170 '**********************************
180 '*  DEFINIEREN VAN DE FUNCTIE-    *
190 '*  TOETSEN F1 TOT EN MET F5.     *
200 '**********************************
210 '
220 KEY 1,"aan"
230 KEY 2,"uit"
240 KEY 3,"links"
250 KEY 4,"boven"
260 KEY 5,"klaar"
270 KEY ON
280 KEY (1) ON:KEY (2) ON:KEY (3) ON: KEY (4) ON:
KEY (5) ON
290 ON KEY GOSUB 450,570,690,770,850
300 '
310 '**********************************
320 '*     OPVRAGEN SPRITE-GROOTTE    *
330 '*      AFDRUKKEN LEGE SPRITE     *
340 '*   WACHTEN OP EEN FUNCTIETOETS  *
350 '**********************************
360 '
370 DIM S$(16,16)
380 GOSUB 1600
390 GOTO 390
400 '
410 '**********************************
420 '*    PIXEL AAN (F1 INGEDRUKT)    *
430 '**********************************
440 '
```

```
450 PRINT CHR$(&HFE)
460 S$(X,Y)="1"
470 X=X+1
480 IF X>S THEN X=1:Y=Y+1
490 IF Y>S THEN Y=1
500 LOCATE X,Y,1
510 RETURN
520 '
530 '**********************************
540 '*     PIXEL UIT (F2 INGEDRUKT)    *
550 '**********************************
560 '
570 PRINT "-"
580 S$(X,Y)="0"
590 X=X+1
600 IF X>S THEN X=1:Y=Y+1
610 IF Y>S THEN Y=1
620 LOCATE X,Y,1
630 RETURN
640 '
650 '**********************************
660 '*      CURSOR NAAR LINKS (F3)     *
670 '**********************************
680 '
690 X=X-1:IF X<1 THEN X=S
700 LOCATE X,Y,1
710 RETURN
720 '
730 '**********************************
740 '*      CURSOR NAAR BOVEN (F4)     *
750 '**********************************
760 '
770 Y=Y-1:IF Y<1 THEN Y=S
780 LOCATE X,Y,1
790 RETURN
800 '
810 '**********************************
820 '*       SPRITE GEDEFINIEERD       *
830 '**********************************
840 '
850 X=0:Y=19:LOCATE X,Y
860 PRINT "einde sprite definitie"
870 INPUT "NAAM VAN DE SPRITE";N$
880 IF S=8 GOTO 960
```

```
890 IF S=16 GOTO 1110
900 END
910 '
920 '********************************
930 '*      SPRITE VAN 8X8 PIXELS      *
940 '********************************
950 '
960 PRINT #1,N$
970 PRINT #1,8
980 FOR I=1 TO 8
990 FOR J=1 TO 8
1000 A=A+ VAL(S$(J,I))*2^(8-J)
1010 NEXT J
1020 PRINT #1,A
1030 A=0
1040 NEXT I
1050 GOTO 1490
1060 '
1070 '*******************************
1080 '*     SPRITE VAN 16X16 PIXELS    *
1090 '*******************************
1100 '
1110 PRINT #1,N$
1120 PRINT #1,16
1130 FOR I=1 TO 8
1140 FOR J=1 TO 8
1150 A=A+VAL(S$(J,I))*2^(8-J)
1160 NEXT J
1170 PRINT #1,A
1180 A=0
1190 NEXT I
1200 FOR I=9 TO 16
1210 FOR J=1 TO 8
1220 A=A+VAL(S$(J,I))*2^(8-J)
1230 NEXT J
1240 PRINT #1,A
1250 A=0
1260 NEXT I
1270 FOR I=1 TO 8
1280 FOR J=9 TO 16
1290 A=A+VAL(S$(J,I))*2^(16-J)
1300 NEXT J
1310 PRINT #1,A
1320 A=0
```

```
1330 NEXT I
1340 FOR I=9 TO 16
1350 FOR J=9 TO 16
1360 A=A+VAL(S$(J,I))*2^(16-J)
1370 NEXT J
1380 PRINT #1,A
1390 A=0
1400 NEXT I
1410 '
1420 '*********************************
1430 '* EEN VOLLEDIGE SPRITE IS NAAR  *
1440 '* SPRFIL GESCHREVEN. VANAF HIER *
1450 '* IS ALLES WEER GELIJK VOOR 8   *
1460 '* EN 16 BITS SPRITES.           *
1470 '*********************************
1480 '
1490 CLS
1500 INPUT "NOG MEER SPRITES DEFINIEREN (J/N)";I$
1510 IF I$<>"j" AND I$<>"J" AND I$<>"N" AND I$<>"n
" GOTO 1500
1520 IF I$="J" OR I$="j" GOTO 1600
1530 PRINT "EINDE PROGRAMMA.     EVEN WACHTEN S.V.
P.":END
1540 '
1550 '*********************************
1560 '* ER IS BESLIST DAT EEN NIEUWE  *
1570 '* ZAL WORDEN GEDEFINIEERD.      *
1580 '*********************************
1590 '
1600 CLS:A$="":A=0
1610 PRINT "SPRITE DEFINITIES:"
1620 PRINT
1630 INPUT "SPRITE-GROOTTE (8/16)";S
1640 IF S<>8 AND S<>16 THEN PRINT "GEEF 8 OF 16 IN
":GOTO 1630
1650 '
1660 '*********************************
1670 '*   INITIALISEREN SPRITE-ARRAY  *
1680 '*********************************
1690 '
1700 FOR I=1 TO 16
1710 FOR J=1 TO 16
1720 S$(I,J)="0"
1730 NEXT J
```

```
1740 NEXT I
1750 '
1760 '********************************
1770 '*      LEGE SPRITE AFDRUKKEN     *
1780 '********************************
1790 '
1800 CLS
1810 FOR I=1 TO S
1820 FOR J=1 TO S
1830 LOCATE J,I:PRINT "-"
1840 NEXT J
1850 NEXT I
1860 X=1:Y=1:LOCATE X,Y,1
1870 RETURN
```

10.2 Verwerken van de sprite file

Het programma uit listing 10.2 leest de sprites uit de cassette file die u met het programma van listing 10.1 had gemaakt. Dit programma dient als voorbeeld van het verwerken van een sprite file. Het doet dan ook niets anders dan het lezen van een sprite, het afdrukken van die sprite op het beeldscherm en het eventueel lezen en afdrukken van volgende sprites. Door de hier gebruikte methode te volgen in uw eigen (spel)programma's, zult u in die programma's probleemloos met sprites kunnen werken. De file met sprites zou u dan of op een aparte cassette of direct achter uw eigen programma moeten plaatsen.

Nadat u is gevraagd de cassette met de sprite file in de recorder te leggen, en die recorder op afspelen te zetten, wordt met regel 160 de cassette file geopend. Daar sprites alleen in de grafische mode kunnen worden weergegeven, zal het grafische beeldscherm ook moeten worden geopend, om er karakters naar toe te kunnen schrijven. Dit wordt met regel 170 gedaan. Het openen van meer dan één file tegelijkertijd is echter niet zonder meer mogelijk. We moeten aan de computer meedelen hoeveel files we tegelijkertijd geopend willen hebben door het statement MAXFILES te geven. Hierop zal de computer zorgen dat er bufferruimtes voor de verschillende files worden gereserveerd.

Nu lezen we (regels 250 tot en met 330) de eerste twee items uit de cassette file. Het eerste item was de naam die we aan de sprite hadden gegeven. Het tweede item was een indicatie van de grootte van de sprite, 8x8 of 16x16. De naam drukken we af op het grafisch beeld-

scherm met regel 290. Afhankelijk van de grootte van de sprite springen we nu naar regel 350 of 480.

In die routines lezen we 8 of 32 volgende items. Steeds wanneer een item is gelezen, kennen we de waarde daarvan toe aan variabele s$ (zie regels 410 en 540). Pas wanneer alle waarden aan s$ zijn toegekend, definiëren we de sprite (zie regels 430 en 560). Hierna wordt s$ leeggemaakt, zodat hij weer kan worden gebruikt voor een eventuele volgende sprite. Nu wordt de sprite op het beeldscherm gezet, met regels 450 en 580.

We hebben nu de naam van de sprite en de sprite zelf op het scherm staan. Regels 670 tot en met 690 stellen ons in de gelegenheid om de sprite rustig te bekijken en om daarna te beslissen of we de volgende sprite willen zien. Zodra we op de RETURN toets drukken wordt de volgende sprite van cassette gelezen en op de hiervoor beschreven wijze op het beeldscherm afgedrukt.

Stel nu echter dat er niet meer sprites op de cassette staan. Dat probleem wordt opgevangen door regel 250. Die regel zegt namelijk wat er moet gebeuren in dat geval. Er moet dan naar regel 760 worden gesprongen. In regel 760 ziet u dat de beide files worden afgesloten, voordat het bericht „EINDE PROGRAMMA" naar het beeldscherm wordt gestuurd.

Listing 10-2

```
10  '*********************************
20  '* MET DIT PROGRAMMA KUNNEN, MET *
30  '* PROGRAMMA SPR1.0 OP CASSETTE  *
40  '* GESCHREVEN SPRITES, WORDEN    *
50  '* INGELEZEN EN AFGEBEELD.       *
60  '*********************************
70  '* OPENEN:                       *
80  '* CASSETTE FILE VOOR INPUT.     *
90  '* GRAFISCH SCHERM VOOR OUTPUT.  *
100 '*********************************
110 '
120 CLS:PRINT "LEG DE CASSETTE MET DE FILE SPRFIL
IN   DE CASSETTERECORDER EN ZET DE RECORDER  OP AF
SPELEN"
130 LOCATE 0,7:PRINT "START PROGRAMMA MET RETURN-T
OETS"
```

```
140 IF INKEY$<>CHR$(13) GOTO 140
150 MAXFILES=2
160 OPEN "CAS:SPRFIL" FOR INPUT AS #1
170 OPEN "GRP:" FOR OUTPUT AS #2
180 N$="":G=0:S=0:S$=""
190 '
200 '**********************************
210 '* LEES EERSTE TWEE ITEMS,         *
220 '*  (NAAM EN SPRITE-GROOTTE)      *
230 '**********************************
240 '
250 IF EOF(1) GOTO 760
260 LINE INPUT #1,N$
270 INPUT #1,G
280 SCREEN 2,2*(G/8)-1
290 PSET(0,20): PRINT #2,N$
300 IF G=8 GOTO 350
310 IF G=16 GOTO 480
320 CLS
330 GOTO 250
340 '
350 '**********************************
360 '*     SPRITE-GROOTTE IS 8*8      *
370 '**********************************
380 '
390 FOR I=1 TO 8
400 INPUT #1,S
410 S$=S$+CHR$(S)
420 NEXT I
430 SPRITE$(1)=S$
440 S$=""
450 PUT SPRITE 1,(120,80)
460 GOTO 670
470 '
480 '*********************************
490 '*     SPRITE-GROOTTE IS 16*16   *
500 '*********************************
510 '
520 FOR I=1 TO 32
530 INPUT #1,S
540 S$=S$+CHR$(S)
550 NEXT I
560 SPRITE$(1)=S$
570 S$=""
```

```
580 PUT SPRITE 1,(120,80)
590 GOTO 670
600 '
610 '**********************************
620 '* DE SPRITE IS AFGEBEELD.         *
630 '* WACHTEN TOT DE OPERATOR DOOR *
640 '* WIL GAAN MET VOLGENDE SPRITE *
650 '**********************************
660 '
670 PSET(0,180)
680 PRINT #2,"RETURN = VOLGENDE SPRITE"
690 IF INKEY$<>CHR$(13) GOTO 690
700 GOTO 250
710 '
720 '**********************************
730 '* ALLE SPRITES ZIJN AFGEBEELD. *
740 '**********************************
750 '
760 CLOSE #1,#2
770 CLS
780 PRINT "EINDE PROGRAMMA"
790 END
```

11. TOVEREN MET CIJFERS

Ongeveer een half jaar geleden kwam mijn dochter thuis met de mededeling dat zij een magisch vierkant kon maken. Ik hoefde maar te zeggen hoeveel hokjes, en zij zou de getallen zo invullen, dat het totaal van alle getallen in een horizontale, verticale of diagonale richting gelijk zou zijn. Dat vond ik erg knap van haar. Maar, ik had wel in de gaten dat ze dat niet uit het hoofd had geleerd.

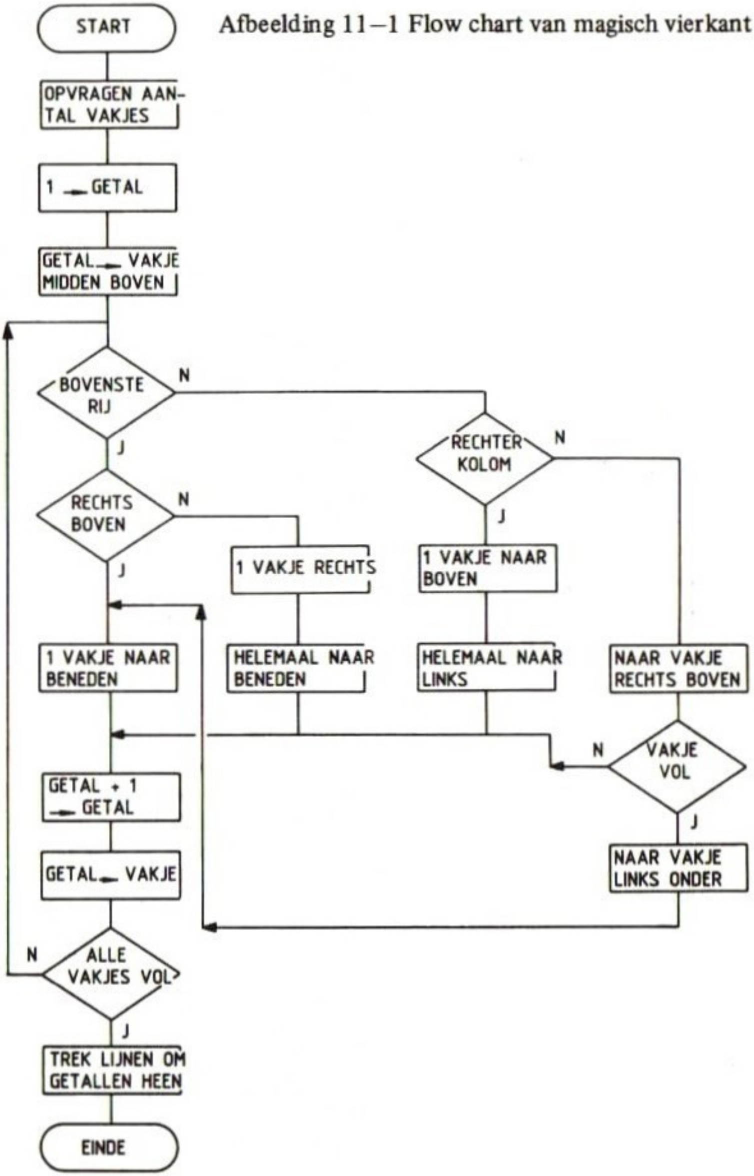
Toen we een paar maanden later op vakantie waren, vroeg ik haar of ze nog steeds van die vierkanten kon maken. „Ja hoor", was haar antwoord, „zeg maar uit hoeveel hokjes dat vierkant moet bestaan". Nu was ik een beetje gemeen, want ik had zelf al uitgevonden dat de manier waarop zij de vierkanten invulde alleen werkte op vierkanten met een oneven aantal hokjes. Daarom vroeg ik haar een vierkant in te vullen met zowel horizontaal als verticaal 4 hokjes. Ze was hevig verontwaardigd dat ze het niet voor elkaar kreeg.

Daarna vroeg ik haar eens een vierkant te proberen met een oneven aantal hokjes. Dit kon ze nog steeds zonder problemen doen. Ik heb haar toen gevraagd hoe ze nu precies te werk ging. En, in feite is de flow chart van afbeelding 11–1 een exacte weergave van haar uitleg.
Die flow chart geeft al zoveel uitleg over het programma, dat ik nu wil volstaan met het geven van enkele aanwijzingen. Zie hiervoor listing 11–1.

Het programma begint te vragen hoeveel hokjes er in horizontale of verticale richting moeten worden gemaakt. Hierop mag u, zoals uit voorgaand relaas duidelijk zal zijn geworden, alleen een oneven aantal hokjes ingeven. Het vierkant wordt dan gedimensioneerd in regel 130. Vervolgens wordt in het midden van de bovenste rij hokjes het cijfer 1 geplaatst (regel 160). Regel 170 vraagt in feite of het laatste cijfer op de bovenste rij werd gezet. Is dit niet het geval dan wordt naar regel 240 gesprongen. Regel 180 vraagt of het laatste cijfer in de rechter bovenhoek werd geschreven. Is dit niet het geval, dan wordt naar regel 280 gesprongen.

Was het laatste getal inderdaad in de rechter bovenhoek geschreven, dan gaan we met regel 190 een vakje naar beneden. We verhogen daarna het getal en schrijven dat in het momentele vakje. Nu kijken we met regel 220 of we alle vakjes hebben gevuld. Is dit zo dan gaan we naar regel 350, waar de PRINT-routine begint waarmee alle cij-

Afbeelding 11–1 Flow chart van magisch vierkant



fers worden afgedrukt.

Stond het laatst geschreven getal niet op de bovenste rij vakjes, dan wordt met regel 240 afgevraagd of het laatste getal soms in de rechter verticale kolom werd geschreven. Is dit niet zo, dan wordt naar regel 310 gesprongen. Op regel 310 gaan we naar het vakje dat rechts boven het laatst geschreven getal staat. Met regel 320 kijken we of daar al een getal in staat. Zo nee, dan schrijven we er het volgende getal in, zo ja, dan gaan we terug naar het vakje waar we vandaan kwamen. Eenmaal teruggekeerd, gaan we met regel 190 een vakje naar beneden, etc.

Indien het laatst geschreven getal niet in de rechter bovenhoek stond, wordt met regel 280 een vakje naar rechts gegaan, om vervolgens met regel 290 helemaal naar beneden te gaan. Dit vakje wordt dan met regel 200 de volgende waarde gegeven.

Ik besef dat het moeilijk is om met woorden te beschrijven wat er precies gebeurt. Doch ik hoop dat deze uitleg u in ieder geval heeft duidelijk gemaakt wat de verschillende BASIC-regels in het programma nu eigenlijk voorstellen. Met behulp van deze uitleg en de flow chart hoop ik dat u in staat zult zijn dit programma te doorgronden.

Enkele opmerkingen tot slot. Indien u alleen een beeldscherm ter beschikking hebt, dan is het maximum aantal hokjes dat de computer voor u kan maken 9. Indien u echter een zogenaamde "Full Line"-printer hebt, dan kunt u ook grotere vierkanten laten maken. Daarbij zult u meer tijd kwijt zijn aan het controleren van de correctheid van de vierkanten, dan de computer aan het maken daarvan.

U ziet dat ik vanaf regel 430 de routine, waarmee een kopie van het beeldscherm naar een printer kan worden geschreven, aan het programma het boegevoegd. Dit stelde mij in staat om het voorbeeld (11.1) te maken. Voor de werking van het programma is deze routine natuurlijk niet nodig. U mag die regels dan ook rustig weglaten. Een beschrijving van de printroutines vidt u in hoofdstuk 6.

Listing 11-1

```
10 '**************************
20 '*  Magische vierkanten   *
30 '**************************
40 '
```

```
50 CLS
60 PRINT "Met dit programma maakt uw MSX-computer"
70 PRINT "magische vierkanten."
80 PRINT "Geef het aantal verticale of"
90 PRINT "horizontale vakjes op (max=9)"
100 PRINT "Aantal moet oneven zijn."
110 INPUT N
120 IF N/2=INT(N/2) GOTO 110
130 DIM V(N*N)
140 GE=1
150 VA=INT(N/2)+1
160 V(VA)=GE
170 IF VA>N GOTO 240
180 IF VA<>N GOTO 280
190 VA=VA+N
200 GE=GE+1
210 V(VA)=GE
220 IF GE>=N*N GOTO 350
230 GOTO 170
240 IF VA/N<>INT(VA/N) GOTO 310
250 VA=VA-N
260 VA=VA+1-N
270 GOTO 200
280 VA=VA+1
290 VA=VA+(N-1)*N
300 GOTO 200
310 VA=VA-N+1
320 IF V(VA)=0 GOTO 200
330 VA=VA+N-1
340 GOTO 190
350 CLS
360 FOR X=0 TO N-1
370 FOR Y=0 TO N-1
380 LOCATE X*3+1,Y*2:PRINT V(X*N+Y+1)
390 NEXT Y
400 NEXT X
410 Y=0
420 '
430 '*******************************
440 '* BEELDSCHERM NAAR LINE PRINTER *
450 '* IN TEKSTMODE 1 (24*40 CHARS). *
460 '*******************************
470 '
480 FOR J=0 TO 21
```

```
490 FOR I=0 TO 39
500 LPRINT CHR$(VPEEK(BASE(0)+J*40+I));
510 NEXT I
520 LPRINT
530 NEXT J
540 END
```

Voorbeeld 11-1

```
Voorbeeld afdrukken magisch vierkant
met de beeldscherm-naar-printer-routine.
```

```
30 38 46 5  13 21 22

39 47 6  14 15 23 31

48 7  8  16 24 32 40

1  9  17 25 33 41 49

10 18 26 34 42 43 2

19 27 35 36 44 3  11

28 29 37 45 4  12 20
```

12. HET GENEREREN VAN EEN TREFWOORDENLIJST

Een trefwoordenlijst is een alfabetische lijst van woorden, met daarachter de nummers van de pagina's waarop die woorden voorkomen. Het is de bedoeling dat alleen die woorden die belangrijk zijn, en waarover iets wordt verteld op de betreffende pagina, in een trefwoordenlijst worden opgenomen. Het zal vaak voorkomen, dat een bepaald woord op meerdere pagina's voorkomt. In dat geval dient het woord slechts eenmaal te worden afgedrukt, gevolgd door alle bijbehorende paginanummers.

Het maken van zo'n trefwoordenlijst geeft een aantal bijzondere problemen.
In die lijst moet het mogelijk zijn om afkortingen of eigennamen die met hoofdletters worden geschreven op te nemen. U hebt echter al uit een der vorige hoofdstukken gezien dat hoofdletters een veel lagere ASCII-code hebben dan kleine letters. Bij het sorteren zouden twee dezelfde woorden, de ene keer met hoofdletters en de andere keer met kleine letters, mijlenver uit elkaar komen te liggen. Dit probleem moet dus worden opgelost. Verder is het zo, dat woorden met dezelfde betekenis niet altijd even consequent geschreven zijn. Een voorbeeld hiervan is het woord microdrive. Misschien staat dat zelfde woord op een andere bladzijde wel geschreven als micro drive met een spatie er tussen. Deze spatie zal er voor zorgen dat het woord tweemaal in de trefwoordenlijst wordt opgenomen, terwijl dat niet de bedoeling is. Dit zal door de mens moeten worden verbeterd, doch het programma moet die mens daartoe wel de gelegenheid bieden.

Het programma uit listing 12–1 is in staat om een trefwoordenlijst te maken. Het biedt het hoofd aan alle hiervoor genoemde problemen. Toch is het niet zo lang als u misschien zou hebben verwacht. Er zitten echter een flink aantal lussen in, speciaal voor het onderscheid tussen hoofd- en kleine letters, zodat er nogal wat instructies moeten worden uitgevoerd voor het maken van een lijstje. We zullen even kort door het programma lopen.

Met de regels 1030 tot en met 1440 kunt u trefwoorden invoeren. De door u ingetypte woorden en paginanummers worden, nadat ze op correctheid zijn gecontroleerd, met de regels 1220 tot en met 1430 op alfabetische volgorde in de trefwoorden array w$ gezet. Voor het zoeken van de juiste plaats binnen de array van een ingetypt karakter

is gebruik gemaakt van de binaire zoekmethode. Er kan echter pas gezocht worden nadat zowel de ingevoerde trefwoorden als de uit de array gelezen trefwoorden beide naar allemaal hoofdletters zijn omgezet. De variabelen a$ en b$ bevatten de hoofdletterversies van de trefwoorden. a$ wordt gevuld met de routine op de regels 1230 tot en met 1260. b$ wordt met de routine op regels 1310 tot en met 1340 gevuld. Daar deze twee routines nogal wat tijd vergen (ca. 1 sec.), was het absoluut noodzakelijk om de juiste plaats in zo weinig mogelijk keren te vinden. Daarom is de binaire zoekmethode in regels 1270 tot en met 1370 toegepast. Regels 1400 tot en met 1420 schuiven alle items uit de array vanaf de plaats waar het nieuwe trefwoord moet komen te staan, een plaatsje naar achteren. Regel 1430 zet dan het nieuwe trefwoord in de lijst.

Door in plaats van een trefwoord een "*" in te geven, weet het programma dat het maken van de trefwoordenlijst is afgelopen. Nu wordt naar regel 1450 gesprongen. Vanaf die regel tot en met regel 1660 wordt u in de gelegenheid gesteld om kleine correcties in de reeds ingegeven trefwoorden aan te brengen. Bent u door de hele trefwoordenlijst heen gelopen, dan worden met de regels 1670 tot het einde alle trefwoorden afgedrukt. Daarbij zorgt regel 1700 er voor dat de woorden niet dubbel worden afgedrukt, doch dat alle paginanummers netjes achter elkaar worden geschreven.

Listing 12-1

```
10 DIM W$(100)
20 CLS
30 WIDTH 40
40 '
1000 REM ***********************
1010 REM * Invoer trefwoorden *
1020 REM ***********************
1021 '
1030 FOR W=1 TO 500
1035 LOCATE 0,20
1040 PRINT "TREFWOORD";W;"?";
1050 LINE INPUT T$
1060 IF LEFT$(T$,1)="*" THEN 1450
1065 LOCATE 0,21
1070 PRINT "PAGINA NR?         ";
1080 LINE INPUT N$
1090 IF LEN(N$)>3 THEN 1080
```

```
1100 FOR A=1 TO LEN(N$)
1110 IF MID$(N$,A,1)<"0" OR MID$(N$,A,1)>"9" GOTO
1070
1120 NEXT
1130 FOR A=1 TO 3-LEN(N$)
1140 LET N$="0"+N$
1150 NEXT A
1160 LET R$=T$
1165 FOR A=1 TO 29-LEN(T$)
1170 R$=R$+" " : NEXT A
1175 R$=R$+N$
1180 LOCATE 0,5
1185 PRINT R$
1188 LOCATE 0,15
1190 PRINT "RETURN=doorgaan, ESC=corrigeren"
1195 I$=INKEY$:IF I$="" GOTO 1195
1200 IF I$=CHR$(27) THEN CLS:GOTO 1035
1210 CLS
1220 IF W=1 THEN W$(W)=R$:NEXT W
1225 A$=""
1230 FOR A=1 TO 32
1235 U$=MID$(R$,A,1)
1240 IF U$>="a" AND U$<="z" THEN U$=CHR$(ASC(U$)-3
2):A$=A$+U$:NEXT A
1250 A$=A$+U$
1260 NEXT A
1270 L=1
1280 R=W
1290 M=INT ((L+R)/2)
1300 IF POKE=R THEN P=L:GOTO 1400
1305 B$=""
1310 FOR A=1 TO 32
1315 U$=MID$(W$(M),A,1)
1320 IF U$>="a" AND U$<="z" THEN U$=CHR$(ASC(U$)-3
2):B$=B$+U$:NEXT A
1330 B$=B$+U$
1340 NEXT A
1350 IF A$<=B$ THEN R=M:GOTO 1290
1360 L=M+1
1370 GOTO 1290
1380 NEXT B
1390 LET P=B
1400 FOR A=W-1 TO P STEP -1
1410 LET W$(A+1)=W$(A)
```

```
1420 NEXT A
1430 LET W$(P)=R$
1440 NEXT W
1450 FOR A=1 TO W-1
1455 LOCATE 0,17
1460 PRINT "record";A
1465 LOCATE 0,18
1470 PRINT W$(A)
1475 LOCATE 0,20
1480 PRINT "RETURN=doorgaan ,ESC=wijzigen"
1485 I$=INKEY$: IF I$="" GOTO 1485
1490 IF I$=CHR$(27) THEN GOSUB 1520
1500 NEXT A
1510 GOTO 1660
1520 INPUT "record nummer ";RN
1530 INPUT "TREFWOORD? ";A$
1540 INPUT "PAGINA NR? ";P$
1550 LET L=LEN(P$)
1560 FOR B=1 TO L
1570 IF MID$(P$,B,1)<"0" OR MID$(P$,B,1)>"9" GOTO
1540
1580 NEXT B
1590 FOR B=1 TO 3-L
1600 LET P$="0"+P$
1610 NEXT B
1620 LET W$(RN)=A$
1625 FOR B=1 TO 29-LEN(A$)
1630 W$(RN)=W$(RN)+" "
1635 NEXT B
1640 LET A=RN-1
1645 W$(RN)=W$(RN)+P$
1648 CLS
1650 RETURN
1660 PRINT "EINDE FILE"
1670 LPRINT
1680 LPRINT LEFT$(W$(1),29);VAL(RIGHT$(W$(1),3));
1690 FOR A=2 TO W-1
1700 IF LEFT$(W$(A-1),29)=LEFT$(W$(A),29) THEN LPR
INT ",";VAL(RIGHT$(W$(A),3));
1710 IF LEFT$(W$(A-1),29)<>LEFT$(W$(A),29) THEN LP
RINT: LPRINT LEFT$(W$(A),29);VAL(RIGHT$(W$(A),3));
1720 NEXT A
1730 END
```

13. BASIC LISTINGS NAAR EEN PRINTER STUREN

Een BASIC listing op het beeldscherm is nooit breder dan 40 karakters. De computer zorgt er zelf voor dat, indien een regel langer dan 40 karakters is, die regel wordt opgesplitst in een aantal stukken van ieder maximaal 40 karakters. Stuur je nu echter diezelfde listing (met ht LLIST commando) naar een full line printer, dan is het resultaat volkomen afhankelijk van de printer. Sommige printers zijn in staat om zelf een Carriage Return en een line Feed tussen te voegen, zodra ze zien dat de regel vol is. Andere printers kunnen dat niet. Sommige printers weten precies wanneer ze in de buurt van de scheurrand van het papier zijn en kunnen dan een aantal regels overslaan. Andere printers kunnen dat niet.

De printer die ik aan mijn MSX computer heb aangesloten, kan, behalve heel mooi afdrukken, bijna niets. Ik had dan ook grote problemen bij het afdrukken van listings waarin BASIC-regels van meer dan bijvoorbeeld 80 karakters voorkwamen. Bovendien ziet mijn printer niet wanneer het papier op is. Bij een lange listing kan het dus wel eens voorkomen dat hij op de rol gaat printen. Om aan al deze problemen een einde te maken heb ik het volgende programma geschreven.

Om te kunnen begrijpen wat het programma doet, is enige kennis vereist van de manier waarop je programma's op een cassette op kunt slaan. Voor normaal gebruik is het commando CSAVE het meest geschikt. Hiermee wordt het hele programma in een keer naar cassette geschreven, en kan het met CLOAD in een keer worden teruggelezen. Het is echter ook mogelijk om het programma met behulp van het commando SAVE "cas:⟨programmanaam⟩" naar cassette te schrijven. In dat geval wordt het programma, zonder dat wij daar erg in hebben, in allemaal korte blokjes naar cassette geschreven. We hebben dan ons programma als het ware in een file op cassette staan.

Om een file van cassette te kunnen lezen, moet die file eerst worden geopend. In het programma ziet u dat gebeuren op regel 160. Nu kunnen we een record uit die file lezen. Dat ziet u op regel 180 gebeuren. Wat lezen we nu? We lezen een string van karakters uit de file. De lengte van die string wordt bepaald door de Carriage Return code. Aan het einde van iedere BASIC-regel staan namelijk een Carriage Return code. Dus, we lezen een hele BASIC-regel, die we in R$ zetten.

Na het inlezen van een BASIC-regel wordt de subroutine die op regel 310 begint, uitgevoerd. Daarin wordt gekeken hoe lang de BASIC-regel is, en indien nodig wordt die regel opgesplitst in een stuk ter lengte van de door u opgegeven regellengte (LR) en een overblijvend stuk van variabele lengte. Hierna wordt het opgegeven aantal letters (LR) afgedrukt. Indien er nog een overblijvend stuk van de regel was, wordt hier weer een stuk ter lengte van LR afgedrukt, etc. etc. Als tenslotte de hele regel is afgedrukt (dit kunnen dus meerdere regels op de printer zijn) wordt uit de subroutine teruggekeerd.

Nu wordt met regel 200 gekeken of het aantal regels dat op de printer is afgedrukt al gelijk is aan of groter dan het opgegeven aantal regels per pagina (RP). Is dit inderdaad het geval, dan wordt de subroutine op regel 500 uitgevoerd. Deze subroutine wacht totdat u op de RETURN-toets hebt gedrukt. Hiermee wordt u in staat gesteld om het papier te verwisselen. Er is immers een hele pagina afgedrukt. Nadat u het papier hebt verwisseld (een nieuw A4 blad er in hebt gedaan) gaat het programma weer verder met het inlezen van BASIC-regels en het afdrukken daarvan op de printer.

Indien u een kettingbaanprinter hebt, hoeft u het papier niet te verwisselen. In dat geval zou u, in plaats van te gaan staan wachten, een routinetje kunnen schrijven die het papier een aantal regels opschuift. Als u weet op welke afstand de scheurranden van elkaar liggen (bijvoorbeeld 66 regels) dan zou u kunnen zeggen dat er steeds 60 regels moeten worden afgedrukt, om daarna 6 regels op te schuiven. Hierdoor is te voorkomen dat u op de scheurrand van het papier gaat printen.

Listing 13-1

```
10 '********************************
20 '* MAAK EEN LISTING OP EEN FULL *
30 '* LINE PRINTER VAN EEN OP CAS- *
40 '* SETTE STAAND PROGRAMMA, MET  *
50 '* OP TE GEVEN REGELLENGTE EN   *
60 '* AANTAL REGELS PER PAGINA.    *
70 '********************************
80 '
90 WIDTH 40:CLS:CLEAR 1000:R=0
100 INPUT "aantal regels per pagina";RP
110 INPUT "aantal letters per regel";LR
```

```
120 LOCATE 0,7:PRINT "LEG DE CASSETTE MET HET TE L
ISTEN PRO-  GRAMMA IN DE CASSETTERECORDER EN ZET D
E RECORDER OP WEERGEVEN."
130 LOCATE 0,12:PRINT "RETURN = programma-start"
140 IF INKEY$<>CHR$(13) GOTO 140
150 CLS
160 OPEN "cas:" FOR INPUT AS #1
170 IF EOF(1)<>0 THEN GOTO 610
180 LINE INPUT #1,R$
190 GOSUB 310
200 IF R=>RP THEN GOSUB 500
210 GOTO 170
220 '
230 '*********************************
240 '* OPSPLITSEN PROGRAMMAREGEL IN *
250 '* AANTAL PRINT-REGELS.         *
260 '* PRINT-REGELS AFDRUKKEN EN    *
270 '* HET AANTAL REGELS BIJHOUDEN  *
280 '* IN VARIABELE R.             *
290 '*********************************
300 '
310 L=LEN(R$)
320 IF L=<LR GOTO 380
330 R1$=LEFT$(R$,LR)
340 R$=RIGHT$(R$,(L-LR))
350 LPRINT R1$
360 R=R+1
370 GOTO 310
380 LPRINT R$
390 R=R+1
400 RETURN
410 '
420 '*********************************
430 '* ER IS EEN VOLLE PAGINA AFGE- *
440 '* DRUKT. OPERATOR KAN EVENTUEEL*
450 '* NIEUW PAPIER IN PRINTER ZET- *
460 '* TEN EN DOORSTARTEN DOOR OP DE*
470 '* RETURN-TOETS TE DRUKKEN.     *
480 '*********************************
490 '
500 LOCATE 10,10
510 PRINT "RETURN = doorgaan"
520 IF INKEY$<>CHR$(13) GOTO 520
530 CLS
```

```
540 R=0
550 RETURN
560 '
570 '******************************
580 '*   EINDE VAN HET PROGRAMMA    *
590 '******************************
600 '
610 CLS
620 PRINT "EINDE LISTING"
630 END
```

ENKELE MSX uitgaven

serie: UW MSX COMPUTER DE BAAS

MSX BASIC HANDBOEK voor iedereen, door A.C.J. Groeneveld
Een kompleet, nederlandstalig handboek voor iedere MSX computergebruiker ISBN 90 6398 100 7

MSX ZAKBOEKJE door Wessel Akkermans
Een vlot geschreven naslagwerk na of naast het handboek. U vindt er o.a. in: niet computergerichte tabellen; de MSX-BASIC instructieset; diverse tabellen die het BASIC-programmeren kunnen versnellen; de Z80 instructieset; hardware-gegevens (connectoren) en een aantal programmaatjes ISBN 90 6398 888 5

MSX DISK HANDBOEK door A.C.J. Groeneveld
Handboek voor diskdrivebezitters om naast het grote handboek te gebruiken. Een zeer volledige behandeling van het disk-gebeuren zelf en de specifieke disk kommando's, uitgebreid met voorbeelden, tabellen en overzichten. Het handboek is aangevuld met interessante programma's, waaronder een tekentafelprogramma en een basisprogramma voor bestandsonderhoud ISBN 90 6398 407 3

MSX QUICK DISK handboek voor iedereen, door A.C.J. Groeneveld
Hèt handboek voor iedere QUICK DISK gebruiker. Uitvoerige behandeling van de sleutelwoorden aangevuld met duidelijke voorbeelden met listing ISBN 90 6398 254 2

SOFTWARE PLUS IN MSX

INTROTAPE MSX, door A.C.J. Groeneveld. Begeleid door instrukties om de computer aan te sluiten en de tape te laden, wordt MSX op een vriendelijke en onderwijzende manier vanuit nul bij de gebruiker geïntroduceerd. Na het doorwerken van deze software is de gebruiker zelf in staat MSX-basic programma's te schrijven
ISBN 90 6398 148 1

MSX-SCRIPT door Ton Weijters
Een menu-gestuurde nederlandstalige tekstverwerker
ISBN 90 6398 189 9

MSX
Praktijkprogramma's 1

Dit eerste deel uit de serie "Praktijkprogramma's voor de MSX computer" is geschreven met als speciaal doel de lezer in staat te stellen zijn eigen toepassingen te programmeren. Hiertoe wordt steeds, waar nodig, eerst een stukje theorie behandeld, onmiddellijk gevolgd door een praktische toepassing daarvan in de vorm van een programma.

De gegeven programma's in MSX, de nieuwe standaard programmeertaal voor hobbycomputers, zijn van uitgebreid commentaar voorzien, zodat u in staat zult zijn de werking ervan te doorgronden, en daardoor de programma's naar eigen behoefte aan te passen.
Bovendien hoopt de auteur, door de keuze van de onderwerpen en de soorten programma's, u ideeën voor eigen programma's aan de hand te doen.

Om u een indruk te geven van de inhoud volgt hier een greep uit de behandelde onderwerpen:

converteren van getallen,
priemgetallen,
ontbinden in factoren,
zoeken en sorteren,
leren typen,
het maken van een trefwoordenlijst,
het definiëren van sprites,
het afdrukken van de beeldscherminhoud
en nog veel meer.

ISBN 90 6398 437 5